VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jyri Liminka

LONWORKS-KENTTÄVÄYLÄN TESTAUKSEN AUTOMATISOINTI

Tekniikka ja liikenne

2011

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Jyri Liminka
Opinnäytetyön nimi	LonWorks-kenttäväylän testauksen automatisointi
Vuosi	2011
Kieli	suomi
Sivumäärä	50
Ohjaaja	Jukka Matila

Tämän opinnäytetyön tavoitteena on suunnitella ja toteuttaa testausympäristö, joka mahdollistaa LonWorks-kenttäväylän automaattisen testauksen ja luoda siihen testitapaukset LonWorksin testispesifikaation mukaisesti. Automaattisen testausjärjestelmän tarkoituksena on nopeuttaa ja helpottaa testausta, sekä parantaa testitulosten luotettavuutta.

Aluksi suunnitellaan toteutustapa testausjärjestelmän rakentamiseen, ja määritetään käytettävät laitteistot ja ohjelmistot alustavan suunnitelman perusteella, joka kuitenkin tarkentuu vielä työn edetessä. Kun laitteistot ja ohjelmistot on saatu toimimaan keskenään yhteistyössä, järjestelmään ohjelmoidaan testitapaukset testispesifikaation mukaisesti. Lopuksi järjestelmän toimivuus testataan ja tarkistetaan, että testitapaukset noudattavat testispesifikaatiota.

Lopputuloksena saadaan LonWorks-kenttäväylälle automaattinen testausjärjestelmä, joka nopeuttaa testausta ja vapauttaa testaajan muihin työtehtäviin testien ajaksi. Tulosten luotettavuus paranee manuaaliseen testaukseen verrattuna, koska nopeuden ansiosta testit voidaan ajaa läpi useita kertoja ja myös inhimillisten virheiden mahdollisuus pienenee huomattavasti.

Työssä perehdytään myös yleisesti kenttäväyliin ja niiden tarjoamiin etuihin perinteisellä tavalla toteutettuihin automaatiojärjestelmiin nähden. Lisäksi tutustutaan testaukseen ja selvitetään tarkemmin automaattisesta testausjärjestelmästä saatavaa hyötyä manuaaliseen testaukseen verrattuna.

Avainsanat LonWorks, kenttäväylä, testaus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Jyri Liminka
Title	Automating the Testing of LonWorks Fieldbus
Year	2011
Language	Finnish
Pages	50
Supervisor	Jukka Matila

The objective of the thesis work is to design and implement a testing environment, which allows automatic testing of LonWorks fieldbus and to create test cases for the system in accordance with the LonWorks test specification. The purpose of the automated testing system is to speed up and ease the testing, as well as improve the reliability of the test results.

At first we conceive a method to implement the testing system and determine the hardware and the software to be used, based on the preliminary plan, which however becomes more precise as the work advances. After the hardware and the software are cooperating, we program the test cases to the system according to the test specification. At last we test that the system works properly and check that the test cases comply with the test specification.

The outcome is an automatic testing system for the LonWorks fieldbus, which speeds up the testing and allows the tester to perform other duties during the testing. The reliability of the results increases compared to manual testing, because the tests can be run through several times thanks to the speed. Moreover, the possibility for human mistakes decreases considerably.

In this thesis, we also take a look at fieldbuses in general and look to the advantages that they offer compared to traditionally implemented automation systems. In addition, we familiarize with testing and find out more precisely, what are the benefits of an automatic testing system in contrast to manual testing.

Keywords LonWorks, fieldbus, testing

SISÄLLYS

TIIVISTELMÄ
ABSTRACT
LYHENNELUETTELO 6
1 JOHDANTO 8
2 TYÖN TARKOITUS JA TAVOITTEET 9
3 AIHEALUE JA TEKNIIKAT 10
3.1 Taajuusmuuttaja 10
3.1.1 Käyttötarkoitus ja käyttökohteet 10
3.1.2 Toimintaperiaate 10
3.2 Kenttäväylät 12
3.2.1 Kenttäväylien käyttö 12
3.2.2 Kenttäväylän valintaperusteet 14
3.2.3 Suosituimpia kenttäväyliä 15
3.2.4 LonWorks 16
3.3 Testaus 19
3.3.1 Testauksen vaiheet 19
3.3.2 Testispesifikaatio 19
3.3.3 Regressiotestaus 20
3.3.4 Mustalaatikkotestaus 20
3.3.5 Kenttäväylien testaus 21
4 LAITTEISTON JA OHJELMISTON MÄÄRITTÄMINEN 23
4.1 Laitteisto 23
4.1.1 LonWorks-optiokortti 23
4.1.2 RS-232-optiokortti 24
4.1.3 LonWorks-verkkosovitin 24
4.1.4 Testikokoonpano 24
4.2 Ohjelmisto 26
4.2.1 NI TestStand 26
4.2.2 Easylon OPC Server 28
4.2.3 DataSocket 29
4.2.4 NI LabVIEW 29

4.2.5 LonMaker Integration Tool....30
5 TESTAUSJÄRJESTELMÄN LUOMINEN 31
5.1 Testausympäristön rakentaminen.... 31
5.1.1 LabVIEW ja TestStand koulutus 31
5.1.2 Suunnittelu ja toteutus....31
5.2 Testitapausten ohjelmointi 37
5.2.1 LonWorksin testispesifikaatio....37
5.2.2 Suunnittelu ja toteutus....38
6 YHTEENVETO JA JOHTOPÄÄTÖKSET 46
LÄHDELUETTELO.... 48

LYHENNELUETTELO

API	Application Programming Interface
ASCII	American Standard Code for Information Interexchange
ATML	Automatic Test Markup Language
CAN	Controller Area Network
COM	Component Object Model
CRC	Cyclic Redundancy Check
DCOM	Distributed Component Object Model
DP	Decentralized Peripherals
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HVAC	Heating, Ventilation and Air Conditioning
Hz	Hertsi, taajuuden yksikkö
I/O	Input/Output
ID	Identification
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
kb/s	Kilobittiä sekunnissa, tiedonsiirtonopeuden yksikkö
LNS	LonWorks Network Service
LonWorks	Local Operating Network

MAC	Media Access Control
Modbus	Modicon bus
NI	National Instruments
NV	Network Variable (verkkomuuttuja)
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OSI	Open Systems Interconnection
PA	Process Automation
Profibus	Process fieldbus
RTU	Remote Terminal Unit
SNVT	Standard Network Variable Type
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
VI	Virtual Instrument
VNI	Virtual Network Interface
XIF	External Interface File
XML	Extensible Markup Language

1 JOHDANTO

Aloitin opinnäytetyön tekemisen loppusyksystä 2010, aiheeksi sain LonWorks-kenttäväylän testauksen automatisoinnin. Toimeksiantajana työlle on Vacon Oyj. Yritys keskittyy kokonaan taajuusmuuttajien kehittämiseen, valmistamiseen ja myyntiin. Vaconin tuotekehitys ja tuotantoyksiköt sijaitsevat Suomessa, Yhdysvalloissa, Kiinassa ja Italiassa. Vaconilla on myyntitoimistoja lisäksi 27 maassa. /1/

Kenttäväylien käyttö mm. teollisuusautomaatiossa on lisääntynyt koko ajan ja suosio kasvaa edelleen. Kenttäväylät ovat kehitetty korvaamaan perinteiset analogiset johdotukset tiedonsiirrossa ja ne tuovat mukanaan monia etuja, joita käsittelen tässä työssä. Vaconin taajuusmuuttajat tukevat optiokorttien kautta useita kenttäväyliä, joista yksi on LonWorks. Ohjelmistoja kehitetään jatkuvasti, jolloin myös testaustarve on suuri ja automaattisella testausjärjestelmällä testausta voidaan tehostaa monella tapaa.

Tämän työn ensimmäinen vaihe on rakentaa LonWorks-kenttäväylän automaattisen testauksen mahdollistava testausympäristö. Työhön kuuluu laitteiston ja ohjelmiston valinta, sekä niiden yhdistäminen toimivaksi järjestelmäksi. Seuraava vaihe on ohjelmoida NI TestStand testienhallintaohjelmistoon testitapaukset, seuraten tarkasti LonWorksin testispesifikaatiota. Työn tarkoituksena on luoda testausjärjestelmä, joka testaa väylän toiminnallisuuden automaattisesti ja testien loputtua antaa tuloksista selkeän raportin.

Aluksi kerron tarkemmin työn tarkoituksesta ja tavoitteista. Luvussa kolme on tietoa taajuusmuuttajasta, jonka jälkeen tarkastelen kenttäväylien toimintaa ja keskityn tarkemmin työn aiheena olevaan LonWorksiin. Seuraavaksi on vuorossa testauksen periaatteita ja eri tekniikoita. Luvussa neljä määrittelen käytettävät laitteistot ja ohjelmistot, jonka jälkeen luvussa viisi käyn läpi yksityiskohtaisesti projektin eri vaiheiden toteuttamisen käytännössä.

2 TYÖN TARKOITUS JA TAVOITTEET

Vaconin taajuusmuuttajat voidaan liittää automaatiojärjestelmään usean eri kenttäväylän kautta. Jokaiselle väylälle on oma optiokorttinsa, joka asennetaan taajuusmuuttajan ohjauskorttiin. Optiokorttien ja ohjauskorttien ohjelmistoja kehitetään koko ajan ja uusia ohjelmistoversioita tulee ulos usein, tämä tarkoittaa myös säännöllistä testausta. Kenttäväylien testaus manuaalisesti on aikaa vievää työtä ja testispesifikaation kunnolliseen läpikäymiseen kuluu helposti kokonainen työpäivä. Automaattisella testausjärjestelmällä voidaan säästää merkittävästi aikaa ja resursseja.

Ensimmäisenä tavoitteena on suunnitella ja toteuttaa ympäristö, joka mahdollistaa kommunikaation tietokoneelle asennetun NI TestStand -testienhallintaohjelmiston ja taajuusmuuttajaan asennetun LonWorks-optiokortin välillä. Tietokoneen ja optiokortin väliin tarvitaan Neuron-sirulla varustettu verkkosovitin, jolla tietokone saadaan toimimaan LonWorks-verkon solmuna. Lisäksi pitää luoda eri ohjelmistojen ja ohjelmoinnin avulla rajapinta, jonka kautta TestStandista pystytään lukemaan ja kirjoittamaan LonWorksin verkkomuuttujia.

Ympäristön luomisen jälkeen toisena tavoitteena on ohjelmoida kaikki testitapaukset TestStandiin LonWorksin testispesifikaatiota noudattaen. Tarkoituksena on saada järjestelmästä mahdollisimman modulaarinen ja selkeä, jolloin se on helposti muokattavissa, kun testispesifikaatiota päivitetään. LonWorks ei ole ainoa testitapauksissa käytettävä väylä, vaan tuloksia tarkistetaan myös HMI-yhteytenä toimivan RS-232-sarjaväylän kautta. Päämääränä on, että automaattinen testaus saadaan käyntiin vaivattomasti ja se ajaa kaikki, tai vain halutut testitapaukset läpi niin monta kertaa kuin on määrätty. Testien loputtua testausraportti generoituu automaattisesti ja siitä tulokset on helposti luettavissa.

3 AIHEALUE JA TEKNIIKAT

3.1 Taajuusmuuttaja

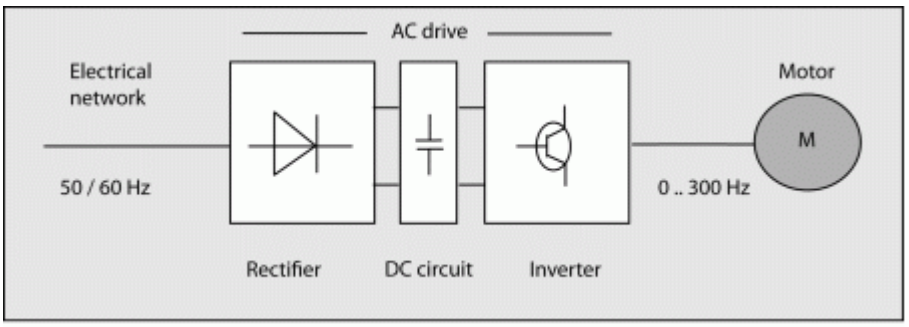
3.1.1 Käyttötarkoitus ja käyttökohteet

Taajuusmuuttajan avulla pystytään säätämään sähkömoottorin pyörimisnopeutta ja vääntömomenttia. Nopeutta säädetään muuttamalla moottorin sisääntulojännitteen taajuutta, tavallisesti välillä 0–300 Hz. Tällä saavutetaan huomattava energiansäästö verrattuna kiinteällä nopeudella pyöriviin moottoreihin, energiaa käytetään ainoastaan sen verran kuin sillä hetkellä tarvitaan. Taajuusmuuttajilla on useita käyttökohteita eri aloilla. Niitä käytetään esimerkiksi ilmastointijärjestelmissä, hisseissä, vedenkäsittelylaitoksissa, öljynporauslautoilla, kaivoksissa ja monissa muissa kohteissa. Taajuusmuuttajat takaavat, että esimerkiksi hissit lähtevät liikkeelle ja pysähtyvät pehmeästi. Usein käyttökohteena ovat tuulettimet ja pumput. Näissä taajuusmuuttajan tuoma energiansäästö on tyypillisesti 20–50%, joten se maksaa itsensä takaisin nopeasti, yleensä alle vuodessa. Taajuusmuuttajan mahdollistama tasainen kiihdytys myös vähentää sähköverkon rasitusta, kun käynnistyksessä syntyvät virtapiikit pienenevät, näin voidaan käyttää pienempiä sulakkeita. Myös äkkinäisistä kiihdytyksistä ja pysäytyksistä johtuva mekaaninen rasitus pienenee. /2/

3.1.2 Toimintaperiaate

Sähkö siirtyy sähköverkosta tasasuuntaajaan, joka muuttaa diodisiltojen avulla vaihtosähkön tasasähköksi. Tasasuuntaaja voi olla yksi- tai kaksisuuntainen. Tasasuunnattu sähkö viedään tasavirtapiirille, jonne energia varastoidaan invertteriä varten. Yleensä sähkö varastoidaan isoihin kondensaattoreihin. Invertteriyksikkö ottaa energiaa tasavirtapiiriltä, muuttaa sen modulaation avulla vaihtosähköksi ja syöttää sitä moottorille. Mitä korkeampi taajuus vaihtosähkölle on valittu, sitä nopeammin moottori pyörii. Yksisuuntaisella tasasuuntaajalla varustettu taajuusmuuttaja ottaa energiaa sähköverkosta ja pyörittää sillä moottoria. Kaksisuuntaisella tasasuuntaajalla varustettu taajuusmuuttaja pystyy myös syöttämään mootto-

rin jarruttaessa syntyvää energiaa takaisin sähköverkkoon, kun mekaaninen energia on ensin muutettu sähköenergiaksi. /2/



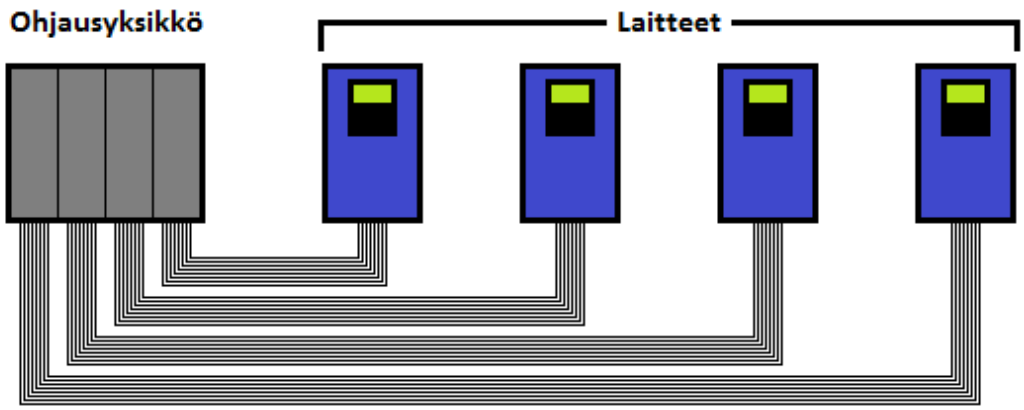
Kuva 1. Taajuusmuuttajan pääkomponentit: tasasuuntaaja, tasavirtapiiri ja invertteri. /2/

3.2 Kenttäväylät

3.2.1 Kenttäväylien käyttö

Kenttäväyliä käytetään mm. teollisuusautomaatiossa laitteiden ohjaamiseen ja monitorointiin. Nykyaikaisissa autoissa voi olla jopa 80 kenttäväylillä toisiinsa liitettyä ohjauslaitetta, jotka kontrolloivat mm. ilmastointia, keskuslukitusta, sytytyksen ajoitusta, jarrujärjestelmää ja luistonestoa. Myös taloautomaatiossa esimerkiksi ilmastoinnin ja hälytysjärjestelmien ohjaus voidaan toteuttaa kenttäväylien avulla. /8; 9/

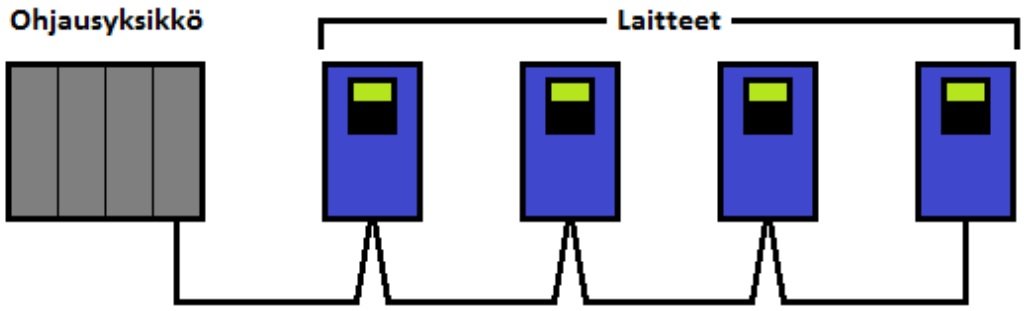
Perinteisesti automaatiojärjestelmässä jokainen laite on liitetty suoraan keskitettyyn ohjausyksikköön analogisesti omilla johdoillaan. Jos järjestelmässä on kymmeniä tai satoja laitteita, ohjausyksiköllе tulevien johtojen määrä on valtava, ja ne vievät paljon tilaa. Johdotuksen asennus- ja huoltokustannukset ovat korkeita, sekä vikatapauksissa vian etsintä on työlästä ja aikaa vievää. Analoginen systeemi on myös paljon alttiimpi häiriöille verrattuna digitaalisesti tapahtuvaan kommunikointiin. /3/



Kuva 2. Perinteinen automaatiojärjestelmä ilman kenttäväylää. Johdot vievät paljon tilaa ja johdotus on hankala asentaa.

Kenttäväylien avulla tiedonsiirto laitteiden, eli solmujen (eng. *node*) kanssa ja niiden välillä voidaan toteuttaa yhdellä kaapelilla täysin digitaalisesti, jolloin tarve monimutkaiselle ja vikaherkälle johdotukselle vähenee tai poistuu kokonaan. Virheenkorjausominaisuudet lisäävät järjestelmän luotettavuutta. Kaapeli menee oh-

jausyksiköstä solmulle, josta se ketjutetaan seuraavalle solmulle ja niin edelleen. Taajuusmuuttajassa kenttäväylän avulla voidaan antaa komentoja, monitoroida arvoja, sekä asettaa ja lukea parametrejä. /3; 4/



Kuva 3. Nykyaikainen kenttäväylällä toteutettu automaatiojärjestelmä. Ohjauskeskuksesta lähtee yksi kaapeli, joka ketjutetaan solmusta toiseen.

Ketjutuksen ansiosta kaikki informaatio on aina kaikkien solmujen saatavilla, eikä ylimääräisiä johtoja solmujen välille tarvita. Asennus, ylläpito ja huolto on helpompaa ja edullisempaa, myös uusien solmujen lisääminen järjestelmään on paljon vaivattomampaa.

On olemassa useita eri väylästandardeja. Moneen järjestelmään riittää yksi kenttäväylä, mutta joissakin tapauksissa voi olla tarvetta pystyä käyttämään yhtä aikaa useita, eri käyttötarkoituksiin parhaiten soveltuvia väyliä. Mikään markkinoilla olevista kenttäväylistä ei ole ylivoimaisesti paras kaikkiin käyttökohteisiin, vaan suosituimmilla väylillä on omat vahvat osa-alueensa. Esimerkiksi Vaconin NX-sarjan taajuusmuuttajissa voidaan käyttää kolmea eri kenttäväylää samanaikaisesti eri tarkoituksiin. Yhtä voidaan käyttää ohjaukseen, toista huoltoon ja kolmatta solmujen väliseen kommunikointiin. Tämän ominaisuuden ansiosta automaatiojärjestelmästä tulee myös paljon joustavampi, koska taajuusmuuttajan avulla voidaan linkittää yhteen eri valmistajien eri kenttäväyliä käyttävät vanhat ja uudet laitteet, jolloin ne saadaan toimimaan yhteistyössä samassa järjestelmässä. /3; 4/

3.2.2 Kenttäväylän valintaperusteet

Kun ollaan valitsemassa kenttäväylää, on tärkeää että sen ominaisuudet sopivat juuri kyseiseen käyttötarkoitukseen. Taloautomaatiossa on erilaiset prioriteetit kuin esimerkiksi prosessiteollisuudessa tai sairaalakäytössä.

Yksi tärkeimmistä huomioon otettavista seikoista on solmujen määrä järjestelmässä, pitää myös osata varautua mahdolliseen solmumäärän kasvuun tulevaisuudessa. Jokaisessa väylästandardissa on käytännön yläraja solmujen määrälle yhdessä väylässä, yleensä muutamista kymmenistä muutamiin satoihin. Monesti on myös mahdollista käyttää toistimia lisäämään solmujen maksimimäärää. Joihinkin Ethernet-pohjaisiin kenttäväyliin voidaan kytkeä teoriassa lähes rajaton määrä solmuja, mutta tiedonsiirto hidastuu solmumäärän kasvaessa. /7/

Väylästandardeilla on eroavaisuuksia myös tiedonsiirtonopeuksissa, maksimietäisyyksissä ja verkon vasteajoissa. Taloautomaatiossa ei ole niin tärkeää saada ilmastointia käynnistymään alle millisekunnissa, tässä tapauksessa muutaman sekunnin vasteaika on täysin hyväksyttävä. Auton jarrujärjestelmässä tai turvatyynyn laukaisussa vasteaika taas on huomattavan tärkeää. Turvatyynystä ei ole paljon hyötyä, jos se laukeaa sekunnin liian myöhään. Tiedonsiirtonopeuden merkitys kasvaa, kun väylässä kiinni olevien solmujen määrä kasvaa tai siirrettävää dataa on paljon. Jos monet väylän solmuista siirtävät dataa yhtä aikaa, voi kaistanleveydestä tulla pullonkaula. Pienessä tilassa, kuten autossa, tiedonsiirtoetäisyyksien rajat eivät tule niin helposti vastaan, mutta isossa tehdashallissa tähän voi joutua kiinnittämään huomiota.

Monet kenttäväylästandardit sallivat vain yhden verkkotopologian käytön, kuten väylä-, tähti-, tai rengastopologia. Topologialla tarkoitetaan tapaa, jolla verkon solmut yhdistetään toisiinsa fyysisesti. Joissain tapauksissa voidaan käyttää myös vapaata topologiaa, joka antaa verkon suunnittelijalle vallan määrätä, mitä kautta solmut on kytketty verkkoon. Huonolla suunnittelulla tämä saattaa kuitenkin hidastaa verkon toimintaa ja vasteaikoja. Esimerkiksi LonWorks on vapaata topologiaa tukeva standardi. /3/

Jotkut väylät mahdollistavat hajautetun ohjauksen, jolloin ei tarvita lainkaan keskitettyä ohjausyksikköä, vaan jokainen solmu voi kommunikoida itsenäisesti suoraan muiden solmujen kanssa. Tämä vähentää viiveitä ja verkon kuormitusta. Keskitetyissä systeemeissä ohjausyksikön vikaantuminen lamaannuttaa koko verkon. Ohjauksen hajauttaminen parantaa vikasietoisuutta. Verkon vasteaika myös usein paranee, kun kaiken informaation ei tarvitse kulkea saman ohjausyksikön kautta, joka voi muodostua pullonkaulaksi. Hajautettu ohjaus ei kuitenkaan sovi kaikkiin järjestelmiin, koska se asettaa tiettyjä vaatimuksia laitteille. Kaikkien verkon solmujen ei tarvitse osallistua ohjaukseen, mutta ainakin osassa laitteista pitää olla enemmän toiminnallisuutta ja ”älyä”, että ne voivat suoriutua ohjaustoimenpiteistä. Jos järjestelmä koostuu pääosin yksinkertaisemmista laitteista, kuten antureista, keskitetty ohjaus voi olla parempi vaihtoehto. Hajautettua ohjausta tarjoaa esimerkiksi LonWorks. /3; 7/

Muita huomioon otettavia tekijöitä kenttäväylän valinnassa ovat mm. mahdollinen olemassa oleva järjestelmä, fyysinen siirtotie, asennuksen vaativuus ja tarjolla olevat työkalut.

3.2.3 Suosituimpia kenttäväyliä

Maailmanlaajuisen suosion saavuttaneisiin kenttäväyliin tällä hetkellä kuuluvat mm. Modbus, Profibus ja CAN. Esittelen nämä kolme seuraavaksi lyhyesti. Tämän työn aiheena olevasta LonWorksista kerron laajemmin luvussa 3.2.4.

Modbus RTU oli ensimmäinen yleisesti hyväksytty kenttäväylästandardi. Se julkistettiin jo 70-luvun lopulla ja on suosittu vielä tänäkin päivänä. Yksi syy suosioon on se, että Modbus on yksinkertainen ja täysin avoin standardi ja sitä voi käyttää kuka tahansa ilman lisenssimaksuja. Modbus RTU perustuu binääriseen tiedonsiirtoon, mutta Modbusista on myös muita versioita. Modbus ASCII siirtää tietoa tekstimuodossa ja Modbus TCP/IP kommunikoi nimensä mukaisesti TCP/IP verkkojen yli. /5; 6/

Saksassa 80-luvun lopulla kehitetty Profibus on myös todella laajalle levinnyt kenttäväylä. Vuoden 2009 lopulla se oli maailman johtava standardi teollisuusau-

tomaatiossa laitteiden määrällä mitattuna. Profibus DP soveltuu laajalle alalle tehdasautomaation eri käyttökohteisiin. Profibus PA on suunniteltu prosessiautomaatiokäyttöön, siinä virta kulkee samaa verkkokaapelia pitkin ja se soveltuu myös räjähdysherkkiin paikkoihin. /10; 11; 12/

CAN-väylä on myös lähtöisin Saksasta, se esiteltiin 1986. Bosch kehitti väylän käytettäväksi autojen tiedonsiirtojärjestelmiin. Nykyään CAN on todella suosittu uusissa autoissa. Se on yleinen autojen lisäksi junissa, laivoissa, lentokoneissa, sairaaloissa ja sotilaskäytössä. Sitä käytetään jonkin verran myös teollisuusautomaatiossa. CAN-väylän virheenkorjaus on huippuluokkaa ja se on hyvin luotettava toiminnaltaan. /9; 13; 14/

3.2.4 LonWorks

LonWorks on Yhdysvaltalaisen Echelonin vuonna 1988 kehittämä automaatiojärjestelmä, joka on suunnattu pääsääntöisesti rakennusautomaation eri käyttökohteisiin. Se soveltuu kuitenkin myös tehdaskäyttöön. LonWorksilla on vahva asema varsinkin Euroopassa. Yhdysvalloissa LonWorksin markkina-asema ei ole niin vahva kuin Euroopassa, mutta se on silti laajalti käytössä ympäri maata. /15; 16/

Rakennusautomaatiossa lämmityksestä, ilmanvaihdosta ja ilmastoinnista käytetään lyhennettä HVAC. Näiden lisäksi rakennusautomaatioon kuuluu mm. valaistuksen, hälytysjärjestelmien ja kulunvalvonnan automatisointi. Myös esimerkiksi sähkön ja veden kulutusta pystytään seuraamaan tarkasti ja näiden tietojen perusteella voidaan yrittää pienentää energiankulutusta automaation avulla. /17/

LonWorks on kokonaisvaltainen automaatioratkaisu, sillä siihen kuuluu mm. kommunikaatioprotokolla, mikroprosessori, lähetin-vastaanottimet (eng. *transceiver*) ja tietokanta. Kaikki mitä tarvitaan automaatiojärjestelmän rakentamiseen on saatavilla Echelonilta ja heidän yhteistyökumppaneiltaan. Echelon tarjoaa verkon suunnitteluun, asennukseen ja ylläpitoon LonMaker-ohjelman. Verkon hallinta sillä onnistuu Echelonin LNS-alustan kautta. LNS-

alusta sisältää verkkopalvelimen ja API:n. LNS API mahdollistaa kolmannen osapuolen ohjelmien kehityksen verkon hallintaan. Echelonilta saa myös LNS Application Developer's Kitin, jolla voi luoda omia verkkotyökaluja. /18; 20/

LonWorks tukee monesta muusta kenttäväylästandardista poiketen peer-to-peer-tekniikkaa, eli verkon solmut pystyvät kommunikoimaan keskenään ilman erillistä ohjausyksikköä. Tämän tekniikan hyötyjä ja haittoja kävin läpi luvussa 3.2.2. On myös mahdollista halutessaan käyttää hierarkista master-slave-verkkoa ja keskitettyä ohjausta. /18/

Järjestelmä perustuu Echelonin kehittämään ja ylläpitämään LonTalk-protokollaan. Se määrittelee kaikki OSI-mallin seitsemän kerrosta. ISO ja IEC ovat myöntäneet LonTalk-protokollalle standardin ISO/IEC 14908-1 ja se on vapaasti sovitettavissa mille tahansa prosessorille. Protokolla ei rajoita tiedonsiirtonopeutta millään tavalla ja voi toimia monella nopeudella, joten siirtotie on vapaasti valittavissa. LonTalk ei tue tiedon salausta, koska monet verkon laitteet voivat tarvita lähetettyä tietoa. Sen sijaan käytetään lähettäjän tunnistusta, että voidaan olla varmoja datan alkuperästä. Protokolla mahdollistaa kommunikoinnin myös IP-verkon kautta. IP-tunneloinnille on myönnetty standardi ISO/IEC 14908-4. /19/

LonWorks mahdollistaa minkä tahansa verkkotopologian käyttämisen. Solmut voi liittää toisiinsa väylä-, tähti-, tai rengastopologian mukaisesti, mutta on mahdollista käyttää myös vapaata topologiaa. Vapaassa topologiassa solmut voi liittää verkkoon mitä kautta haluaa. Myös siirtotie on vapaasti valittavissa, joskin standardisoituja ovat vain kierretyn parikaapelin ja sähköverkon käyttö siirtotienä. Kahden edellä mainitun lisäksi voidaan käyttää esimerkiksi radiotaajuuksia, infrapunaa, koaksiaalikaapelia tai optista kaapelia. Uusi-Seelantilainen yhtiö on käyttänyt jopa sähköistettyä aitaa LonWorks-tiedonsiirtoon, kehitettyään siihen tarkoitukseen sopivan lähetin-vastaanottimen. /18; 19/

Koko järjestelmän keskipisteessä on Neuron siru, joka sisältää koko LonTalk protokollan, kolme prosessoria, muistin, I/O-nastat, tiedonsiirtoportin, laiteohjelman ja käyttöjärjestelmän. Media Access -prosessori hoitaa viestien lähettämisen, vas-

taanottamisen ja CRC-tarkistuksen, Network-prosessori hallinnoi verkkoliikennettä, kuten reitityksiä ja uudelleenlähetyksiä ja Application-prosessori ajaa käyttäjän sovellusta. Jos Neuronin 8-bittinen Application-prosessori ei riitä käyttäjän muokkaaman ohjelman ajamiseen, voidaan protokolla sovittaa tehokkaammalle prosessorille ja käyttää Neuronia vain huolehtimaan kommunikaatiosta. Jokaisella Neuron sirulla on oma uniikki 48-bittinen Neuron ID, jolla ne yksilöidään. Tämä vastaa MAC-osoitetta, jota käytetään mm. tietokoneiden verkkokorteissa. /19/

Solmujen välinen kommunikaatio tapahtuu käyttäen verkkomuuttujia (NV), jotka on määritelty jokaiselle verkon solmulle. NV voi olla joko tulo- (eng. *input*) tai lähtömuuttuja (eng. *output*). Echelon ylläpitää listaa erilaisista standardeista muuttujien tyypeistä, joita kutsutaan nimellä SNVT. SNVT voi olla lähes mikä tahansa fysikaalinen suure tai esimerkiksi kytkimen tila. Mahdollisia yksiköitä muuttujan arvolle ovat mm. watti, tosi/epätosi, kilogramma, desibeli, celsiusaste, ASCII-merkki, prosentti ja niin edelleen. NV:t voidaan myös sitoa (eng. *bind*) toisiinsa. Solmun lähtö voi olla sidottuna yhteen tai useampaan tuloon, jotka ovat sen omia, tai samassa verkossa olevien toisten solmujen tuloja, kunhan ne ovat samaa tyyppiä. Aina kun lähdön arvo muuttuu, se päivittyy myös jokaiseen siihen sidottuun tuloon samanaikaisesti. /19/

LonMark-järjestö ylläpitää toiminnallisia profiileja (functional profiles) erilaisille laitteille. Nämä profiilit määrittävät kullekin laitetyypille käytettävät NV:t ja muun toiminnallisuuden sovelluskerroksella. Toimintojen standardisointi mahdollistaa eri valmistajien LonWorks-laitteiden toimimisen samassa järjestelmässä. Vaconin taajuusmuuttajissa LonWorks-yhteyksiin käytettävä OPT-C4-optiokortti tukee LonMarkin Variable Speed Motor Drive -profiilia. /30/

3.3 Testaus

3.3.1 Testauksen vaiheet

Ohjelmistotestaus sisältää kolme päävaihetta: Yksikkötestaus, integraatiotestaus ja systeemitestaus. Yksikkö- tai komponenttitestauksessa testataan pienimpien ohjelman osien toimintaa, nämä voivat olla esimerkiksi yksittäisiä funktiota tai metodeita. Yksikkötestauksen tekee yleensä ohjelmoija itse, joskus myös erillinen testaaja. Integraatiotestaus suoritetaan yksikkötestauksen jälkeen, siinä testataan yksikkötestauksen läpäisseiden ohjelman osien välistä yhteistoimintaa, kuten funktiokutsuja. Projektin koosta riippuen, integraatiotestauksen voi suorittaa ohjelman kehittäjä tai ulkopuolinen taho. Viimeisenä suoritetaan järjestelmätestaus, jossa testataan koko ohjelmisto painottaen pääosin toiminnallista testausta rakenteellisen testauksen sijaan. Tämä suoritetaan yleensä mustalaatikkomallin mukaan erillisen testaajan toimesta, joka tietää miten koko systeemin tulisi toimia, mutta ei sen sisältämiä funktioita tai muita sisäisiä rakenteita. Tässä työssä keskitytään nimenomaan järjestelmätestaukseen. Kun taajuusmuuttajan optiokortille tai ohjauskortille tulee uusi ohjelmistoversio, se on jo läpäissyt rakenteelliset testit, ennen kuin kenttäväylän standardin mukaista toiminnallisuutta lähdetään testaamaan. /32; 33; 34; 35/

3.3.2 Testispesifikaatio

Yhteensopivuuden takaamiseksi kenttäväylän toiminnan pitää noudattaa määritettyä toimintamallia tai standardia. Väylän käyttäytyminen pitää olla täysin ennustettavissa kaikissa mahdollisissa käyttötilanteissa ja erikoistapauksissa. Aina kun tietyt ehdot täyttyvät, lopputuloksen pitää olla sama. Esimerkiksi annettu komento asettaa aina moottorin pyörimään tiettyä nopeutta, tai laitteen mennessä vikatilaan, kenttäväylältä pitää tulla aina oikea vikailmoitus. Standardin pohjalta luodaan testispesifikaatio, jossa määritellään kaikki kenttäväylälle tehtävät testit, jotka sen pitää läpäistä ennen käyttöönottoa.

Testispesifikaatio sisältää testitapauksia, joiden avulla pyritään simuloimaan kaikki mahdolliset skenaariot, mitkä voivat toteutua väylällä varustettua laitetta käy-

tettäessä. Testitapaus määrittelee väylälle annettavat syötteet ja odotetut vasteet. Testitapaukset tulisi suunnitella niin, että jonkin niistä epäonnistuessa virhe pystytään paikantamaan nopeasti. Jos liian monta toimintoa testataan samaan aikaan, ei välttämättä ole selvää, miksi testitapaus ei mennyt läpi. Kuitenkin yhteistoiminnan testausta varten, pitää toistensa kanssa vuorovaikutuksessa olevia toimintoja sisällyttää samaan testitapaukseen. Testispesifikaation suunnittelu vaatii järjestelmällisyyttä. Testitapausten tulee olla selkeitä, mutta liika yksinkertaistaminen saattaa kasvattaa niiden määrän liian suureksi, jolloin koko spesifikaation selkeys kärsii. /37/

3.3.3 Regressiotestaus

Regressio tulee englannin sanasta regression, joka tarkoittaa taantumista. Kun aiemmin testattuun ja toimivaksi todettuun ohjelmakoodiin tehdään muutoksia, esimerkiksi lisätään uusia ominaisuuksia tai korjataan virheitä, ne voivat vaikuttaa myös muiden, jo testattujen ohjelman osien toimintaan. Jos muutokset saavat ennestään toimivan ohjelman osan toimimaan virheellisesti, ohjelma on regressoitunut, eli taantunut kehityksessä taaksepäin. Regressiotestauksella pyritään havaitsemaan ohjelman regressoituminen mahdollisimman nopeasti, jolloin se pystytään korjaamaan heti. /21/

Regressiotestaus tapahtuu käytännössä siten, että jokaisen koodimuutoksen jälkeen ohjelman kaikki osat testataan kokonaisuudessaan uudelleen ja joka kerta samoilla testitapauksilla kuin ennenkin. Ohjelman kehittyessä mahdolliset uudet testitapaukset lisätään vanhojen jatkoksi. Virheen löytyessä sen syy on helposti paikannettavissa ja korjattavissa, kun sen tiedetään johtuvan viimeisimmästä muutoksesta. Kun regressiotestauksessa havaittu virhe on korjattu, pitää regressiotestaus suorittaa taas täydellisesti uudelleen, koska korjauksessa on voinut syntyä uusia virheitä. /21/

3.3.4 Mustalaatikkotestaus

Mustalaatikkotestauksella (eng. *black box testing*) testataan yleensä ohjelmiston tai järjestelmän toiminnallisuutta kokonaisuutena. Testaaja ei tiedä järjestelmän

sisäistä rakennetta, kuten ohjelman sisältämiä funktioita, jolloin testattava järjestelmä on hänelle ”musta laatikko”. Mustalaatikkotestauksessa käytetään toiminnallisen spesifikaation pohjalta luotuja testitapauksia. Järjestelmään annetaan syötteitä ja järjestelmän antamia vasteita verrataan testispesifikaatiossa määriteltyihin tuloksiin. Testitapauksia tarvitaan usein paljon ja niiden rakentamiseen kuluu aikaa, mutta rakentaminen voidaan aloittaa heti kun spesifikaatio on julkaistu, eikä ohjelmiston valmistumista tarvitse odottaa. Testitapaukset eivät ota kantaa järjestelmän rakenteeseen tai toteutustapaan, mustalaatikkotestauksessa ainoastaan toiminnallisuus ratkaisee. /31/

Eräs mustalaatikkotestauksessa käytettävä menetelmä on raja-arvoanalyysi. Mikäli standardissa on määritelty esimerkiksi jonkin muuttujan minimiarvoksi 1 ja maksimiarvoksi 100, ei testata kaikkia mahdollisia arvoja, vaan keskitytään raja-arvojen ympäristöön. Käytännössä tässä tapauksessa testattaisiin arvoilla 0, 1, 100, 101, sekä yhdellä muulla arvolla väliltä 2–99. Yleensä virheellinen toiminta tapahtuu juuri raja-arvojen läheisyydessä, joten tällä tavalla saadaan testistä riittävän kattava, testaamatta jokaista arvoa erikseen. /36/

3.3.5 Kenttäväylien testaus

Vaconilla kenttäväylän regressiotestaus suoritetaan aina, kun kyseisen kenttäväylän tarjoavalle optiokortille tulee uusi ohjelmaversio. Kun taajuusmuuttajan ohjauskortille tulee uusi ohjelmaversio, testataan kaikki kenttäväylät joiden toimintaan muutokset voivat vaikuttaa. Kyse on mustalaatikkotestauksesta, koska testataan koko järjestelmän toiminnallisuutta, eikä testaajan tarvitse olla perillä optiokortin tai ohjauskortin ohjelmistojen rakenteista.

Kenttäväylien testausta varten on rakennettu testikokoonpano, jossa on sähkömoottoriin kytketty taajuusmuuttaja. Taajuusmuuttajan I/O-nastat on kytketty USB-sovittimen kautta tietokoneeseen. Tietokoneelta voidaan I/O:n kautta mm. käynnistää ja sammuttaa laite, sekä aiheuttaa vikatiloja. Taajuusmuuttajaan asennetaan testattavan kenttäväylän tarjoava optiokortti ja se liitetään tietokoneeseen siihen sopivalla kaapelilla, tarvittaessa jonkin sovittimen kautta. Tietokoneella on tarvittavat ohjelmistot kenttäväylien testaamiseen.

Jos kenttäväylälle ei ole automaattista testausjärjestelmää, pitää työntekijän seurata testispesifikaatiota kohta kohdalta ja käydä testitapaukset läpi yksitellen. Arvojen kirjoittaminen väylälle tapahtuu manuaalisesti. Väylältä tulevat arvot luetaan siihen soveltuvasta ohjelmasta ja mahdolliset poikkeamat kirjataan ylös. Näin yhden väylän testaamiseen kuluu aikaa yleensä koko päivä ja työntekijä on sidottuna siihen koko tämän ajan. On myös mahdollista, että testauksessa tapahtuu inhimillinen virhe, jonka takia testin lopputulos ei ole täysin luotettava. Esimerkiksi väsymys tai kiire voi aiheuttaa sen, että jotain testitapausta ei käydä läpi oikeaoppisesti tai se jää kokonaan pois. Vaikka testitapaus suoritettaisiin oikein, mahdollinen virhe voi jäädä silti huomaamatta.

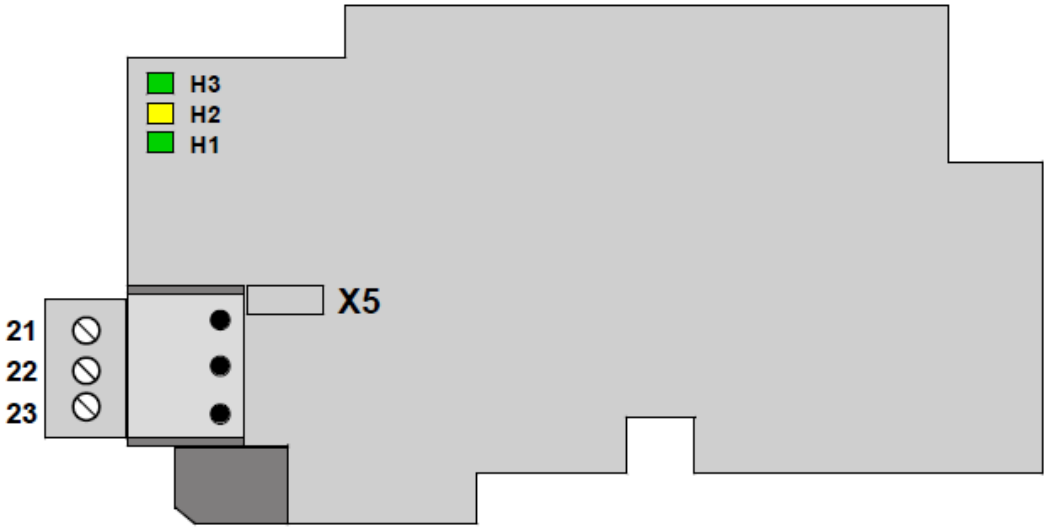
Mikäli automaattinen testausjärjestelmä kenttäväylälle on olemassa, työntekijä laittaa testit pyörimään, jonka jälkeen hän vapautuu muihin tehtäviin siksi aikaa, kun testit ovat kesken. Kun testaus on valmis, hän katsoo automaattisesti generoidusta raportista kuinka testit onnistuivat. Automaattisen testausjärjestelmän nopeuden ansiosta testit ehditään ajamaan läpi yleensä kymmeniä kertoja siinä ajassa, kun testispesifikaatio on käyty läpi yhden kerran manuaalisesti. Mitä useammin testit ajetaan, sitä luotettavampia tulokset ovat. Testit voidaan laittaa ajoon vaikka iltapäivällä töistä lähtiessä ja aamulla töihin tullessa nähdään raportista tulokset. Automatisointi vähentää merkittävästi inhimillisten virheiden mahdollisuutta, jolloin testien tulokset ovat luotettavampia. On kuitenkin ehdottoman tärkeää, että automaattisessa testausjärjestelmässä ei ole virheitä. Tämän takia se pitää käydä läpi tarkasti useamman henkilön toimesta ennen järjestelmän käyttöönottoa. Sama pätee, kun testausjärjestelmään lisätään uusia testitapauksia, tai muokataan vanhoja testitapauksia vastaamaan päivitettyä testispesifikaatiota.

4 LAITTEISTON JA OHJELMISTON MÄÄRITTÄMINEN

4.1 Laitteisto

4.1.1 LonWorks-optiokortti

Optiokortti OPT-C4 on varustettu lähetin-vastaanottimella FT-X1, joka sallii kaikkien LonWorksin tukemien verkkotopologioiden käytön. Siirtotienä käytetään kierrettyä parikaapelia ja tällä saavutetaan 78 kb/s siirtonopeus. Yhteen verkkosegmenttiin voidaan liittää enintään 64 solmua, verkkosegmentit voidaan liittää toisiinsa reitittimen avulla. /22/



Kuva 4. Optiokortti OPT-C4. /22/

Optiokortissa on 3-napainen irrotettava ruuviliitin kaapelille. Navat 21–22 ovat dataa varten ja napaan 23 kytketään maadoitus. Jumpperilla X5 voidaan valita, käytetäänkö yhtä vai kahta terminointia, valinta riippuu verkon topologiasta. Ledeistä H3 näyttää Neuron-sirun tilan, H2 optiokortin tilan ja H1 kenttäväylämoduulin tilan. /22/

4.1.2 RS-232-optiokortti

Taajuusmuuttajan HMI-väylänä toimii RS-232-sarjayhteys, jonka tarjoaa optiokortti OPT-D3. Myös taajuusmuuttajan ohjauspaneeli käyttää RS-232-yhteyttä, joten väylän kautta pystytään tekemään kaikki samat asiat kuin suoraan ohjauspaneelilta, ja kaikki paneelilta luettavissa olevat arvot ja parametrit ovat luettavissa myös väylän kautta. HMI-väylää käytetään automaattisessa testauksessa mm. ohjaustavan valitsemiseen I/O:n ja kenttäväylän välillä, sekä kenttäväylältä luettujen arvojen tarkistamiseen.

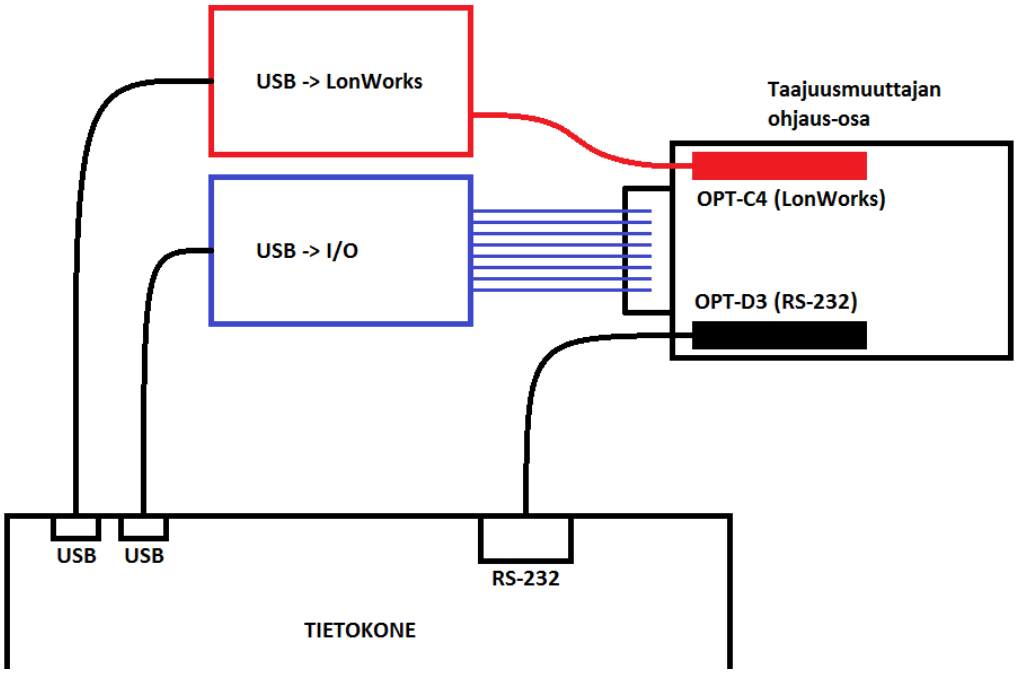
4.1.3 LonWorks-verkkosovitin

Tietokoneen ja optiokortin välille piti löytää verkkosovitin, jolla tietokone saataisiin liitettyä osaksi LonWorks-verkkoa ja se näkyisi taajuusmuuttajalle verkon solmuna. Tähän tarkoitukseen sopiva laite löytyi saksalaiselta Gesyteciltä. Verkkosovittimen malli on Easylon VNI Interface USB, se on pienikokoinen ja ottaa myös käyttöjännitteen suoraan USB-väylästä, joten erillistä virransyöttöä ei tarvita. Laite tukee USB 2.0 ja USB 1.1 tiedonsiirtostandardeja. Verkkosovittimessa on 3-napainen ruuviliitin parikaapelille ja RJ-45-liitin. /23/

Yksi syy Gesytecin valintaan verkkosovittimen toimittajaksi oli se, että heillä oli tarjolla myös OPC-palvelin LonWorksille. Näin voitiin varmistaa laitteiston ja ohjelmiston yhteensopivuus. OPC-palvelimesta kerron tarkemmin luvussa 4.2.2.

4.1.4 Testikokoonpano

Testikokoonpanoon kuuluu sähkömoottoriin liitetty Vacon 100 -sarjan taajuusmuuttaja, jonka ohjauskortille on asennettu optiokortit OPT-C4 LonWorks yhteydelle ja OPT-D3 RS-232-sarjayhteydelle. Gesytecin LonWorks-verkkosovitin on kytketty OPT-C4-optiokorttiin parikaapelilla ja tietokoneeseen USB-kaapelilla. OPT-D3 on kytketty sarjakaapelilla tietokoneen sarjaporttiin. Taajuusmuuttajan I/O-nastat ovat kytketty tietokoneeseen USB-muuntimen kautta.



Kuva 5. Yksinkertaistettu kuva testikokoonpanon kytkennöistä.

4.2 Ohjelmisto

4.2.1 NI TestStand

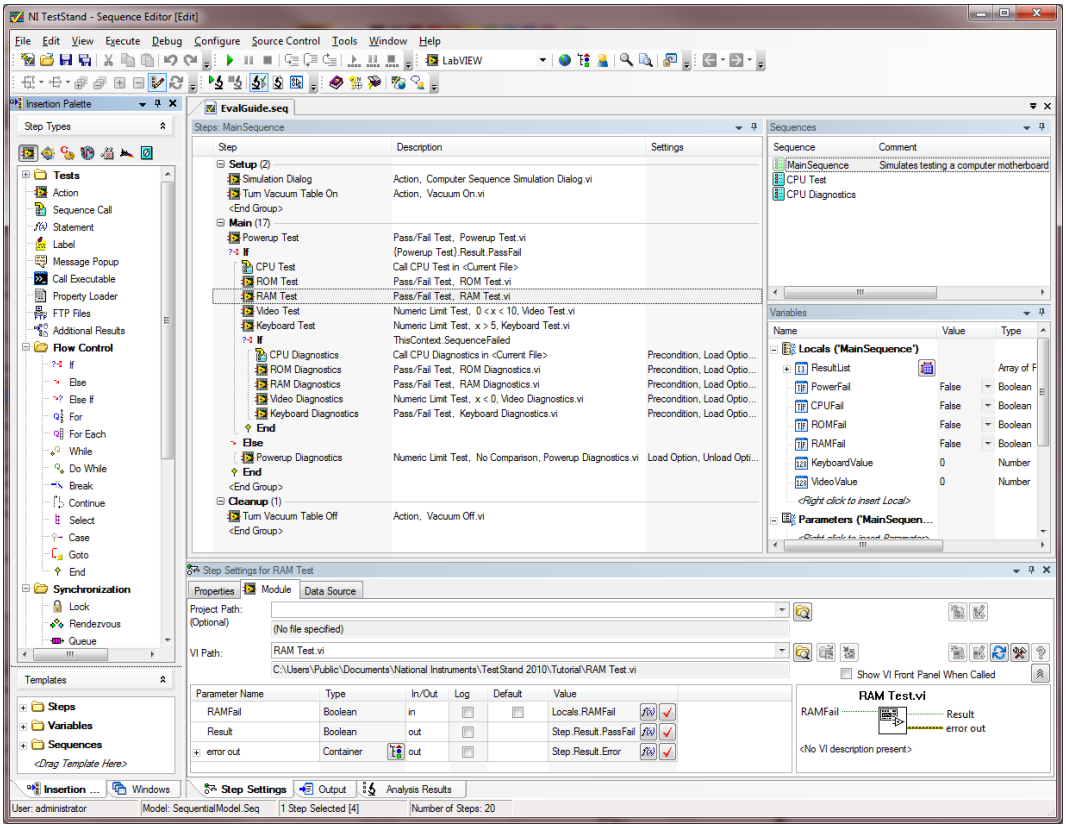
NI TestStand on Vaconilla käytettävä testienhallintaohjelmisto. Se on suunniteltu nimenomaan testauksen automatisointia varten ja sillä pystyy toteuttamaan hyvin monipuolisia ja joustavia testikokonaisuuksia. TestStandissa käsitellään sekvenssitiedostoja. Sekvenssitiedosto sisältää main-sekvenssin ja mahdollisesti alisekvenssejä. Main-sekvenssistä voidaan kutsua alisekvenssejä, jotka voivat sijaita myös toisessa sekvenssitiedostossa. Testisekvenssien sisältä voidaan kutsua lähes millä tahansa ohjelmointikielellä tehtyjä ohjelmamoduuleita halutuilla parametreillä, paluuarvot saadaan TestStandiin, kun moduuli on suoritettu. Testausraportti generoituu testien loputtua automaattisesti ja sen ulkoasu on muokattavissa halutun näköiseksi. Vaihtoehdot raportin muodolle ovat XML-, HTML- ja ATML-dokumentti, sekä ASCII-tekstitiedosto.

Testisekvenssit muodostuvat stepeistä eli askeleista, jotka suoritetaan järjestyksessä ylhäältä alas. Seuraavassa muutama esimerkki TestStandin sisäänrakennetuista step-tyypeistä:

- String Value Test, (Multiple) Numeric Limit Test, Pass/Fail Test: Erilaiset testityypit kutsuvat ohjelmamoduulia ja vertaavat paluuarvoja määritettyihin raja-arvoihin. Tietotyyppi voi olla numeroarvo, merkkijono, tai tosi/epätosi. Testin lopputulos on joko *Pass* tai *Fail*.
- Action: Ei testaa mitään, vaan kutsuu ohjelmamoduulia ja suorittaa halutun toiminnon. Normaalisti lopputuloksena on joko *Done* tai *Error*.
- Sequence Call: Kutsuu testisekvenssiä, joka voi sijaita myös toisessa sekvenssitiedostossa.
- Statement: Suorittaa halutun ohjelmakoodin. Esimerkiksi suorittaa laskutoimituksen ja tallentaa tuloksen muuttujaan.
- Call Executable: Käynnistää ohjelman halutuilla parametreillä, suorittaa skriptin tai käyttöjärjestelmäkomennon.
- If, for, while, ym.: Testisekvenssien sisällä voidaan käyttää yleisiä ohjelmointirakenteita.

- Wait, semaphore, queue ym.: Erilaisia synkronisaatioon ja ajoitukseen käytettäviä step-tyyppejä.

Näiden lisäksi TestStandissa on monia muita step-tyyppejä esimerkiksi tietokantojen käsittelyyn ja jokaiseen stepiin voi sisällyttää ohjelmakoodia. Valmiita step-tyyppejä voi muokata tai tehdä itse kokonaan uusia. Testien kehittäjä voi myös luoda testisekvensseille oman hallintapaneelin nappeineen, ledeineen ja graafeineen, jolla voidaan helpottaa ja selkeyttää loppukäyttäjän työtä.



Kuva 6. NI TestStand käyttöliittymä. Avattuna oleva sekvenssitiedosto on TestStandin mukana tuleva esimerkki.

4.2.2 Easylon OPC Server

OPC on joukko standardeja, jotka määrittävät rajapinnan automaatiolaitteiden ja tietokoneohjelmistojen välille. Ensimmäinen ja tärkein näistä standardeista on Data Access Specification, se syntyi automaatioyhtiöiden ja Microsoftin yhteistyön tuloksena vuonna 1996. Standardeja kehittää ja ylläpitää OPC Foundation. Tiedonsiirto perustuu Microsoftin OLE COM- ja DCOM -teknologioihin, joten OPC toimii vain Windows-käyttöjärjestelmien päällä, se noudattaa palvelin/asiakas-toimintamallia. /24/

OPC-palvelin on tietokoneohjelma, joka tukee yhtä tai useampaa OPC-standardia. OPC-asiakasohjelmien ja palvelimien välinen kommunikaatio (lukeminen ja kirjoittaminen) tapahtuu aina standardin mukaisesti, joten eri OPC-asiakasohjelmat pystyvät kommunikoimaan saman palvelimen kanssa. OPC-standardista on kuitenkin eri versioita, joidenkin valmistajien palvelimet ovat taaksepäin yhteensopivia ja toisten eivät. OPC-palvelimien ja laitteiden välinen kommunikaatiotapa riippuu laitteen käyttämästä protokollasta. Esimerkiksi LonWorksille tarvitaan erilainen OPC-palvelin kuin Modbusille, koska protokollat eivät ole yhteensopivia. OPC-palvelin toimii käytännössä erillisten laiteajureiden korvaajana, asiakasohjelmien ei tarvitse tietää laitteiden käyttämää protokollaa voidakseen kommunikoida niiden kanssa, muuten jokainen ohjelma tarvitsisi omanlaisensa ajurin. /25; 26/

Tiedonsiirtoon optiokortin ja tietokoneen välillä vaikutti järkevimmältä ratkaisulta käyttää OPC-palvelinta. NI TestStand ei kuitenkaan tue suoraan OPC-tageja, joten ensin piti keksiä keino kommunikaation toteuttamiseen NI TestStandin ja OPC-palvelimen välillä. Tämä keino oli DataSocket, josta kerrotaan tarkemmin luvussa 4.2.3. Gesyteciltä tilattiin LonWorksille tehty Easylon OPC Server, koska heiltä löytyi myös sopiva LonWorks-verkkosovitin. Yhteensopivuuden kannalta oli hyvä, että laitteisto ja ohjelmisto saatiin samalta toimittajalta.

4.2.3 DataSocket

Yksi pohdinnan aiheista työn määrittely- ja suunnitteluvaiheessa oli löytää tapa, jolla NI TestStandista käsin voidaan lukea ja kirjoittaa OPC-palvelimen tageja. Alustavan tutkimuksen jälkeen arvojen välitykseen parhaiten soveltuvaa rajapintaa tiedusteltiin National Instrumentsilta. Eri vaihtoehtojen punnitsemisen tuloksena DataSocket vaikutti sopivimmalta ratkaisulta ohjelmien välisen kommunikaation toteuttamiseen.

DataSocket on teknologia, joka on suunniteltu yksinkertaistamaan reaaliaikaisen datan jakaminen ja julkaiseminen automaatio- ja mittaussovellusten välillä, jotka sijaitsevat samalla koneella tai verkossa. Se sisältyy mm. NI LabVIEW -kehitysympäristöön. Datan lukeminen ja kirjoittaminen tapahtuu URL:ien avulla, jotka koostuvat protokollasta, koneen nimestä ja datan nimestä. DataSocket valittiin tähän työhön juuri siksi, että se tukee OPC-protokollaa ja voi siten toimia OPC-asiakasohjelmana. /27/

4.2.4 NI LabVIEW

NI LabVIEW on ohjelmointiympäristö, jossa käytetään graafista G-ohjelmointikieltä. Ohjelmakoodissa käytetään tekstin sijasta kuvakkeita, jotka on yhdistetty toisiinsa johdoilla. Esimerkiksi silmukka rakentuu vetämällä hiirellä kuvakkeita silmukan sisälle ja asettamalla ehto silmukasta poistumiselle. LabVIEW:ssa funktioita kutsutaan VI:ksi. Kehitystyökalut ovat tarjolla Windows-, Linux- ja Mac-ympäristöille. LabVIEW:lla tehdyt ohjelmat toimivat näiden lisäksi reaaliaikakäyttöjärjestelmillä. Ohjelmien siirto alustalta toiselle onnistuu yleensä ilman suuria muutoksia ohjelmakoodiin, koska suorat käyttöjärjestelmäkutsut ovat harvinaisia. /28/

VI koostuu block diagramista ja front panelista. Front panel sisältää VI:n käyttöliittymän ja block diagram itse koodin. Kun front paneliin lisätään esimerkiksi numeroilmaisin, se näkyy myös block diagramin puolella ja sen voi kytkeä johdolla vaikka näyttämään laskutoimituksen lopputuloksen. Johto vain vedetään hiirellä kuvakkeiden välille block diagramissa.

Tässä työssä LabVIEW:ia tarvittiin sen mukana tulevan DataSocket-teknologian takia. LabVIEW:sta löytyi valmiit komponentit DataSocketia varten, joilla pystyi rakentamaan tarvittavat VI:t. Mukana tuli myös valmiita esimerkkejä DataSocketin käyttöön.

4.2.5 LonMaker Integration Tool

Echelonin LonMaker on ohjelmisto LonWorks-verkon suunnitteluun, luomiseen ja ylläpitoon. Verkon suunnittelun käyttöliittymänä toimii Microsoft Visio, joka hyödyntää LonMakerin sisältämää LNS-palvelinta. Suunnittelu tapahtuu graafisesti, käyttämällä LonMakerin mukana tulleita valmiita lohkoja LonWorks-verkoille, tarvittaessa voi luoda myös omia. Lohkot vedetään hiirellä paikoilleen ja määritetään niille tulot ja lähdöt. Sitominen, eli tulojen ja lähtöjen liittäminen toisiinsa tapahtuu yksinkertaisesti piirtämällä johto niiden välille. LonMaker oli jo Vaconilla käytössä ennestään, joten sitä ei tarvinnut hankkia erikseen. /29/

5 TESTAUSJÄRJESTELMÄN LUOMINEN

5.1 Testausympäristön rakentaminen

5.1.1 LabVIEW ja TestStand koulutus

Vaconilta tarjottiin minulle aivan työn alkuvaiheessa mahdollisuus osallistua National Instrumentsin kurssille, jossa perehdytään LabVIEW:n ja TestStandin käyttöön. Kurssi oli hyvin tiivis, yhden viikon aikana käytiin läpi laaja asiakokonaisuus. Sain kurssilta hyvän pohjan opinnäytetyötä varten, ja uskon että ilman kurssia työn alkuvaiheessa olisi mennyt huomattavasti enemmän aikaa ohjelmien peruskäytön opiskeluun.

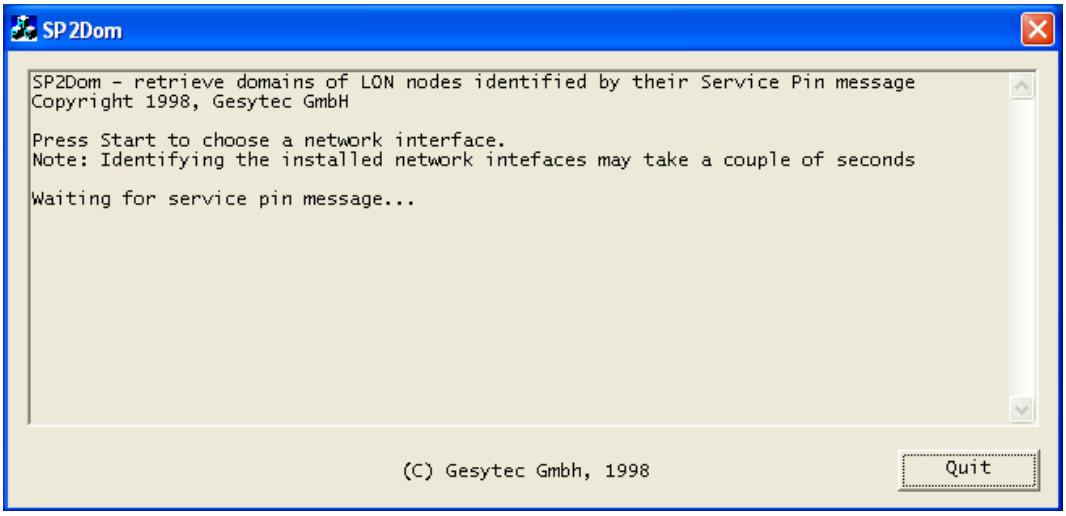
5.1.2 Suunnittelu ja toteutus

Suuri osa suunnittelusta tehtiin jo laitteiston ja ohjelmiston määritysvaiheessa, koska ennen tilausten tekoa piti tietää, että työ pystytään toteuttamaan näillä työkaluilla. Suunnitellessani projektin toteutustapaa, keskustelin asiasta ohjaajani ja muun henkilöstön kanssa. Näissä keskusteluissa tuli esille mahdollinen OPC-palvelimen käyttö optiokortin kanssa kommunikointiin, ja asiaa tutkittuani totesin sen parhaimmaksi vaihtoehdoksi. TestStandin ja OPC-palvelimen välisen yhteyden toteutuksesta sain muutamia ehdotuksia National Instrumentsilta, lopulta päädyin käyttämään tähän DataSocketia, joka on integroitu LabVIEW-kehitysympäristöön.

Aluksi LonWorks-verkon NV:t pitää saada näkymään OPC-palvelimella, eli tietokoneeseen liitetty verkkosovitin asennetaan verkon solmuksi. LabVIEW:lla ohjelmoidaan DataSocketia varten VI:t, jotka pystyvät toimimaan OPC-asiakasohjelman roolissa, eli kirjoittamaan OPC-tageihin ja lukemaan niitä. VI:t otetaan käyttöön TestStandiin sen sisältämän LabVIEW-adapterin avulla, jolloin TestStandista pystytään kommunikoimaan OPC-tagien, eli tässä tapauksessa LonWorks-verkkomuuttujien kanssa.

Ensimmäinen vaihe oli saada OPC-palvelimen ja optiokortin välinen yhteys toimimaan. Koneelle asennettiin Gesytecin Easylon OPC Serverin versio 2.2. Kun

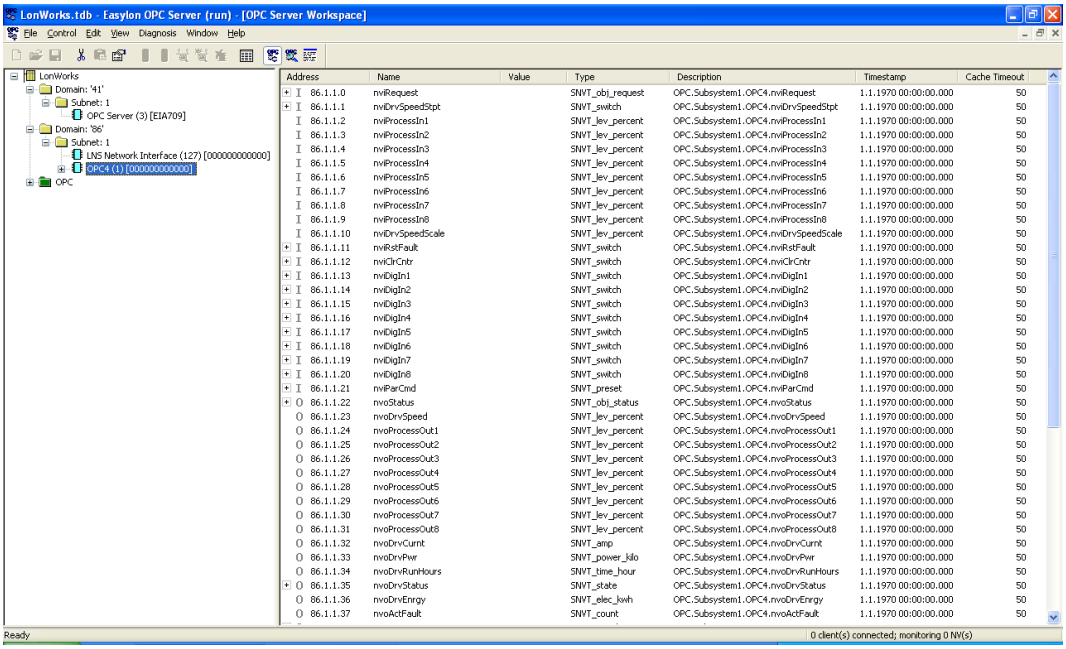
palvelin oli asennettu, taajuusmuuttajan optiokortti piti liittää verkkoon LonMakerin avulla, jonka jälkeen OPC-palvelin pystyi tunnistamaan sen. Tunnistaminen tapahtui käynnistämällä palvelimen mukana tullut apuohjelma nimeltä determine domains ja aktivoimalla service pin taajuusmuuttajasta. Toimenpide antaa taajuusmuuttajalle osoitteen LonWorks-verkossa, osoite koostuu toimialueesta (eng. *domain*), aliverkosta (eng. *subnet*) ja solmun ID:stä (eng. *node-ID*).



Kuva 7. Determine domains. Taajuusmuuttaja saa osoitteen LonWorks-verkosta, kun sen service pin aktivoidaan.

OPC-palvelin käynnistettiin config-tilassa ja määritettiin sen asetuksiin käytettävä verkkosovitin. Seuraavaksi asetuksiin täytyi antaa nimi palvelimen käyttämälle tietokannalle, johon tallennetaan tiedot jokaisesta verkon NV:stä. Palvelimelle annettiin myös domain, subnet ja node-ID. Domain pitää olla sama kuin optiokortilla, mutta subnet ja node-ID eivät saa olla samoja, tai optiokorttia ei enää löydy. Sen jälkeen skannattiin väylän laitteet, jolloin palvelin löysi taajuusmuuttajan ja näytti sen sisältämät NV:t. Tietokantaan piti vielä lisätä nimet kaikille verkkomuuttujille. Tietokanta avattiin Microsoft Accessilla, muuttujien nimet saatiin optiokortin xif-tiedostosta. Käynnistämällä OPC-palvelin varmistettiin, että nimet olivat päivittyneet sinne. Palvelimelta pystyi suoraan kirjoittamaan arvoja muuttujiin, ilman erillistä asiakasohjelmaa. Toimintaa testattiin mm. pyörittämällä moottoria väylän kautta ja tarkkailemalla kuinka arvot

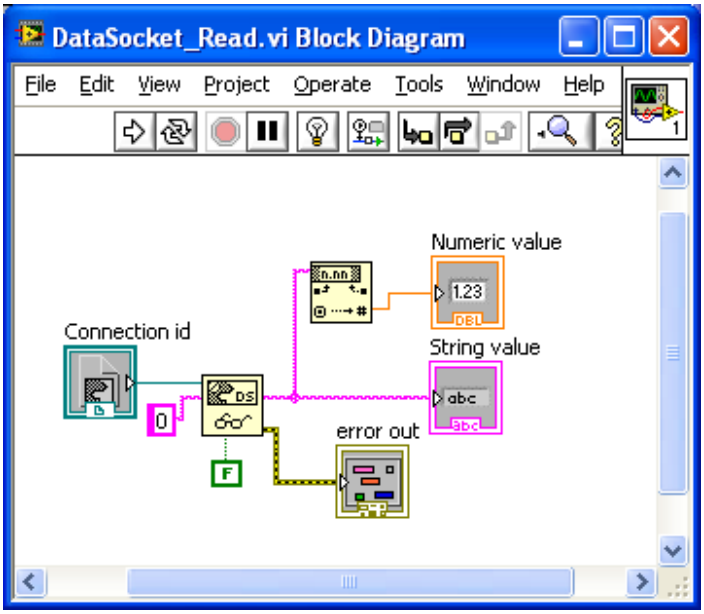
muuttuvat. Samaa kokeiltiin myös ilmaisella Softing OPC Demo Clientilla, sekin toimi odotetulla tavalla.



Kuva 8. Easylon OPC Serverin käyttöliittymä run-tilassa, jokaisella optiokortin NV:llä on oma URL-osoite.

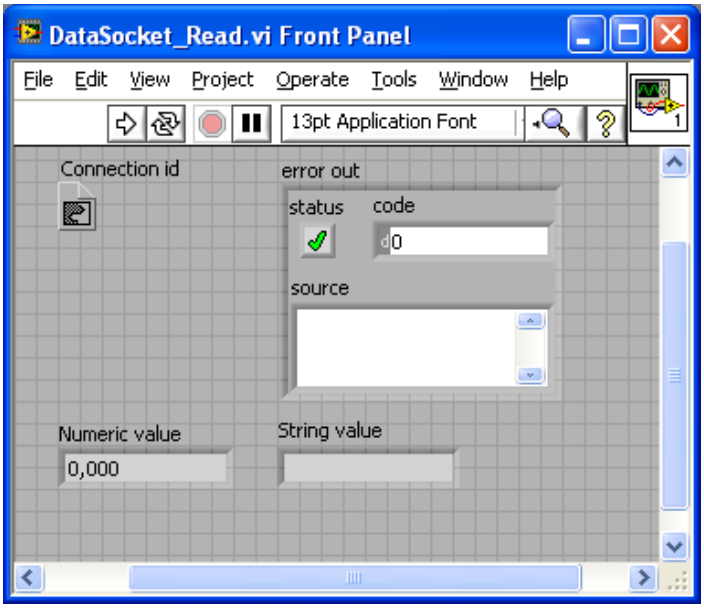
Seuraava vaihe oli ohjelmoida LabVIEW:lla VI:t DataSocketia varten, että OPC-palvelimen ja TestStandin välinen yhteys saataisiin toimimaan. Yhteyden avaamiselle ja sulkemiselle, sekä verkkomuuttujien lukemiselle ja kirjoittamiselle tarvittiin kaikille omat VI:t.

DataSocket Read VI:ta käytetään NV:n arvon lukemiseen OPC-palvelimelta DataSocketin kautta. Toimintaperiaate on seuraava: Kun DataSocket-yhteys on avattu DataSocket Open VI:llä, se antaa ulos connection ID:n yhteydelle. Connection ID vaihtuu joka kerta, kun yhteys avataan, joten se pitää ottaa talteen esim. taulukkoon. DataSocket Read VI saa Connection ID:n tästä taulukosta, jonka jälkeen se osaa lukea oikean NV:n arvon. Kaikki arvot tulevat OPC-palvelimelta string-muotoisina, mutta double-muunnos lisättiin, että arvot saadaan ulos myös numeerisina ja niitä on mahdollista käyttää suoraan laskutoimituksissa.



Kuva 9. DataSocket Read VI, block diagram.

Front panelin oikeassa yläkulmassa on VI:n kuvake, josta saa näkyviin myös liittimen I/O-nastoineen. Jokainen tulo ja lähtö pitää olla kytkettynä liittimeen, että niitä voi käyttää TestStandissa. Liittäminen tehdään vetämällä johto front panelin komponentista liittimen terminaaliin.



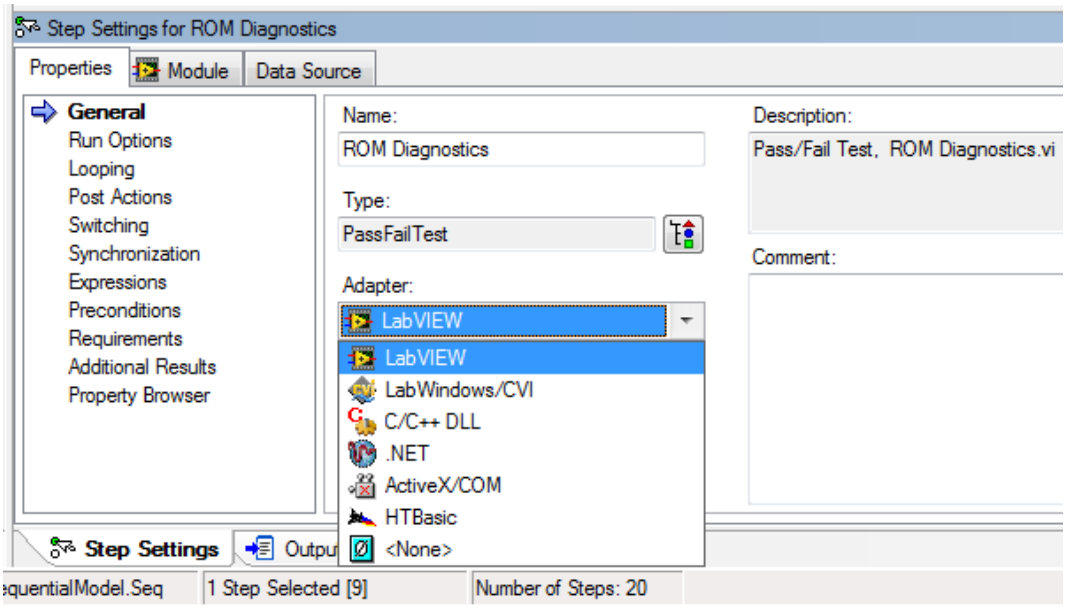
Kuva 10. DataSocket Read VI, front panel.

DataSocket Write VI toimii samalla periaatteella, eli se saa connection ID:n taulukosta, johon DataSocket Open VI on sen tallettanut ja kirjoittaa arvon string-muotoisena verkkomuuttujaan. Testisekvenssin lopussa DataSocket Close VI sulkee yhteydet käymällä koko taulukon läpi for-silmukassa.

DataSocket Open VI:ssä voidaan valita avataanko yhteys luku-, vai kirjoitustilassa. Yhteys pitäisi olla mahdollista avata myös ReadWrite-tilassa, jota käytetään sellaisille muuttujille, jotka toimivat sekä tuloina että lähtöinä. Myöhemmin testitapauksia ohjelmoidessa kävi kuitenkin ilmi, että ReadWrite-tila ei toiminut oikein, vaan se jumitti testin, kun verkkomuuttujaan yritettiin kirjoittaa. Tämä oli ollut bugi aiemmassa LabVIEW:n versiossa, mutta se piti olla korjattu. Asiaa selvitettiin National Instrumentsin kanssa, mutta ratkaisua ongelmaan ei keksitty.

Koska asiaa ei saatu selvitettyä kohtuullisessa ajassa, otettiin käyttöön vaihtoehtoinen ratkaisu. Kun luku ja kirjoitusyhteys samaan verkkomuuttujaan eivät voineet olla auki yhtä aikaa, ainoaksi vaihtoehdoksi jäi ohjelmoida vielä tällaista käyttöä varten omat VI:t, joissa yhteys avataan ennen luku- tai kirjoitusoperaatiota ja suljetaan heti sen jälkeen. Normaalisti on parempi, että yhteydet avataan aina testitapauksen alussa, tehdään tarvittavat operaatiot ja suljetaan yhteydet vasta testitapauksen lopuksi. Esimerkiksi moottorin pyörimisnopeus voidaan lukea useaan kertaan saman testitapauksen sisällä, jolloin yhteyden avaaminen ja sulkeminen joka kerta veisi turhaan aikaa. Onneksi näitä tulo/lähtö-muuttujia ei ole kuin muutama ja niiden testaus ei ole aikakriittistä, joten lisäviiveestä ei koitunut suurempia ongelmia.

TestStandissa VI:t otetaan käyttöön niin, että stepin ominaisuuksista valitaan käyttöön LabVIEW-adapteri. Stepin moduuliasetuksista löytyy kohta VI path, johon tulee halutun VI:n hakemistopolku. Polun voi määrittää joko suhteelliseksi tai absoluuttiseksi.



Kuva 11. Adapterin valinta TestStandissa ja eri vaihtoehdot.

Lopuksi koko järjestelmän toiminta voitiin testata yksinkertaisella sekvenssillä, jossa kirjoitetaan moottorille käyntikomento tietyllä nopeudella, jonka jälkeen luetaan pyörimisnopeus ja verrataan sitä kirjoitettuun arvoon. Kun VI:t oli tehty ja TestStandista pystyttiin kirjoittamaan ja lukemaan optiokortin verkkomuuttujia OPC-palvelimen kautta, testausympäristö oli valmis ja voitiin siirtyä seuraavaan vaiheeseen.

5.2 Testitapausten ohjelmointi

5.2.1 LonWorksin testispesifikaatio

Sain päivitetyn testispesifikaation käyttööni, kun se oli ensin katselmoitu Vaconin toimesta. Spesifikaatio sisältää 13 testitapausta, joilla testataan LonWorks-väylän toimintaa eri tilanteissa. Kaksi näistä sisältää useampia testejä, joten käytännössä testitapauksia on 24. Testitapaukset on esitelty alla lyhyesti.

- **Power Failure Test:** Testataan, palautuuko optiokortti kunnolla virtakatkoksesta.
- **Heartbeat Test:** Testataan receive heartbeatin toiminnallisuutta. Taajuusmuuttajan pitää reagoida, jos tietyt NV:t eivät päivity annetun ajan sisällä.
- **NviDrvSpeedScale & NviDrvSpeedStpt Test:** Kirjoitetaan eri arvoja muuttujiin nviDrvSpeedScale ja nviDrvSpeedStpt ja tarkistetaan että muuttujien nvoDrvSpeed ja nvoDrvStatus arvot ovat oikeita.
- **Process Data Write/Read:** PD out 1–8 asetetaan HMI:n kautta näyttämään suoraan PD in 1–8 arvoja. LonWorksilla kirjoitetaan arvoja sisääntuloihin ja tarkistetaan että ulostulojen arvot vastaavat niitä.
- **Parameter Read/Write:** Kirjoitetaan parametreja ja tarkistetaan menivätkö ne perille. Sama testi kuudelle eri ID:lle.
- **Counter Reset Test:** Testataan trippilaskureiden nollausta.
- **NciDrvSpeedScale Test:** Tarkistetaan että NV:n nciDrvSpeedScale arvo säilyy samana taajuusmuuttajan uudelleenkäynnistyksen jälkeen.
- **Digital Input Test:** Kirjoitetaan digitaaliset tulot 1–8 ykkösiksi järjestyksessä ja joka välissä tarkistetaan HMI:n kautta Control Wordin arvo.
- **Digital Output Test:** Testataan digitaaliset lähdöt 1–8.
- **NciMaxSpeed & NciMinSpeed Test:** Luetaan maksimi- ja minimitaajuudet, verrataan niitä maksimi- ja miniminopeuksista sekä nimellistaajuudesta laskemalla saatuihin arvoihin.
- **Network Configuration variable test:** Testataan useiden asetusten tekoon käytettävien muuttujien toiminnallisuutta.

- **Send Hearbeat Test:** Testataan send heartbeatin toiminnallisuutta. Send heartbeat määrittää maksimiajan, jonka kuluttua tietyt muuttujat päivittyvät automattisesti.
- **NviRequest Test:** Testataan seitsemän eri komennon toimintaa. Jokainen testi eroaa toisistaan, toisin kuin Parameter Read/Write testissä, jossa eri ID:t testataan samalla tavalla. Komennot ovat:
    - RQ_NORMAL
    - RQ_UPDATE_STATUS
    - RQ_CLEAR_STATUS
    - RQ_REPORT_MASK
    - RQ_DISABLED
    - RQ_ENABLE
    - RQ_CLEAR_ALARM

5.2.2 Suunnittelu ja toteutus

Suunnitelmana oli luoda TestStandin sekvenssitiedostoon oma sekvenssi jokaiselle testitapaukselle ja alitestitapaukselle, sekä kutsua näitä main-sekvenssistä järjestyksessä. Lisäksi tarvittiin omat sekvenssit ennen LonWorks-testien aloittamista suoritettaville toimenpiteille, kuten taajuusmuuttajan käynnistämiselle. Nämä muidenkin kenttäväylien testaamisessa tarvittavat sekvenssit olivat kuitenkin jo olemassa ennestään, joten niitä voitiin pienen muokkauksen jälkeen käyttää myös tässä työssä.

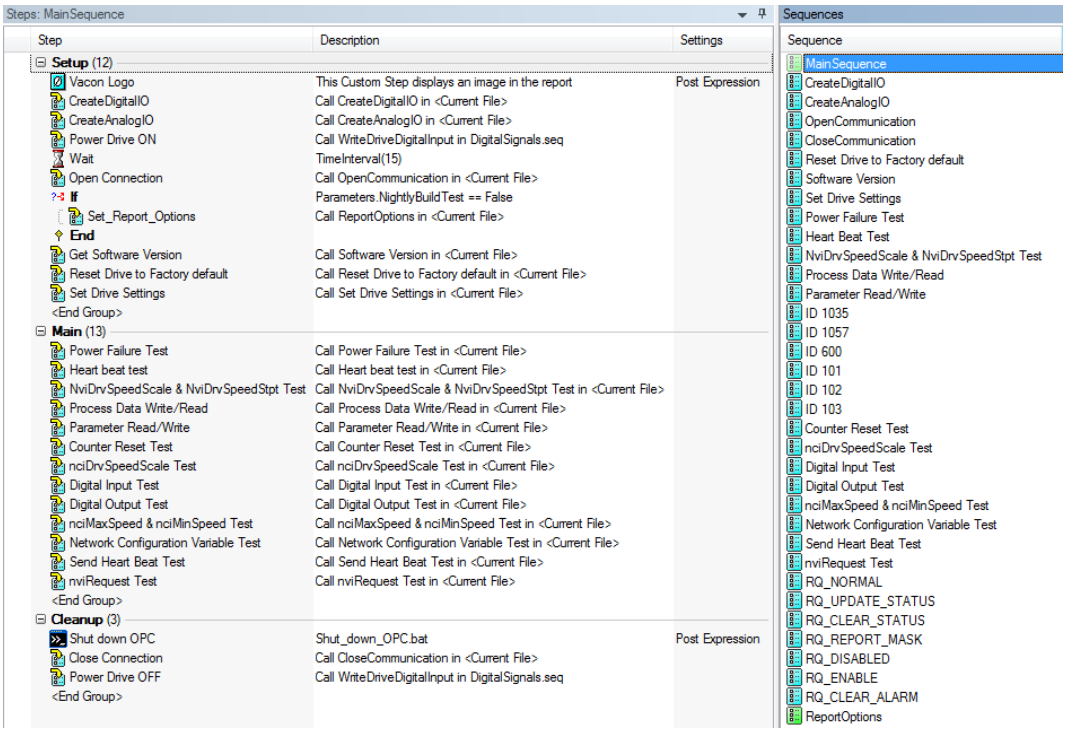
Jo suunnitteluvaiheessa arveltiin, että send heartbeatin testaaminen saattaa tuottaa ongelmia, koska se vaatii muutaman NV:n sitomista toisiinsa ja tässä vaiheessa ei vielä ollut selvää, kuinka sitominen onnistuu OPC-palvelinta käytettäessä. Päätettiin että tehdään kaikki muut ensin ja keskitytään send heartbeatin toteuttamiseen työn lopussa.

Ennen kun LonWorks-testit voidaan aloittaa, pitää testausjärjestelmä alustaa käyttökuntoon. Tämä suoritetaan automaattisesti main-sekvenssin alussa. Tarvittavat

sekvenssit olivat jo olemassa ja ne saatiin pienten muokkausten jälkeen toimimaan myös tässä työssä, joten niitä ei tarvinnut lähteä luomaan tyhjästä. Alustuksessa suoritetaan seuraavat toimenpiteet:

- Luodaan I/O-yhteydet. I/O:n kautta mm. käynnistetään ja sammutetaan taajuusmuuttaja, sekä luodaan vikatiloja.
- Käynnistetään taajuusmuuttaja I/O:n kautta.
- Avataan RS-232-sarjayhteys HMI:lle.
- Asetetaan raporttiasetukset.
- Luetaan ohjelmistoversio.
- Resetoidaan taajuusmuuttajaan tehdasasetukset.
- Asetetaan taajuusmuuttajaan oikeat asetukset testausta varten.

Alustuksen jälkeen LonWorks-testit käydään läpi samassa järjestyksessä kuin ne ovat testispesifikaatiossa, lopuksi suljetaan yhteydet ja sammutetaan taajuusmuuttaja.



Kuva 12. Main-sekvenssi TestStandissa.

Eri testitapaukset erosivat toisistaan huomattavasti ohjelmoinnin helppouden kannalta. Osa testitapauksista oli aika suoraviivaisia ja siten yksinkertaisia toteuttaa TestStandiin, osa taas oli monimutkaisempia ja niitä ohjelmoidessa kului enemmän aikaa toteutustavan miettimiseen.

Seuraavaksi kerrotaan tarkemmin Parameter Read/Write testistä ja sen toteuttamisesta TestStandiin. Testi tehdään kuudelle eri ID:lle, käytän esimerkkinä ID:tä 1035. Testitapauksen kuvaus spesifikaatiossa sisältää käytettävät NV:t ja esimerkit luku- ja kirjoituskomennoista. Testispesifikaatiossa määritelty testaustapa on lyhyesti seuraava:

1. Lue parametrin arvo (oletusarvo).
2. Kirjoita uusi arvo, oletusarvo + 1.
3. Lue arvo, arvon pitää olla oletusarvo + 1.
4. Tarkista arvo myös paneelilta tai HMI:n kautta.
5. Palauta parametrin oletusarvo.

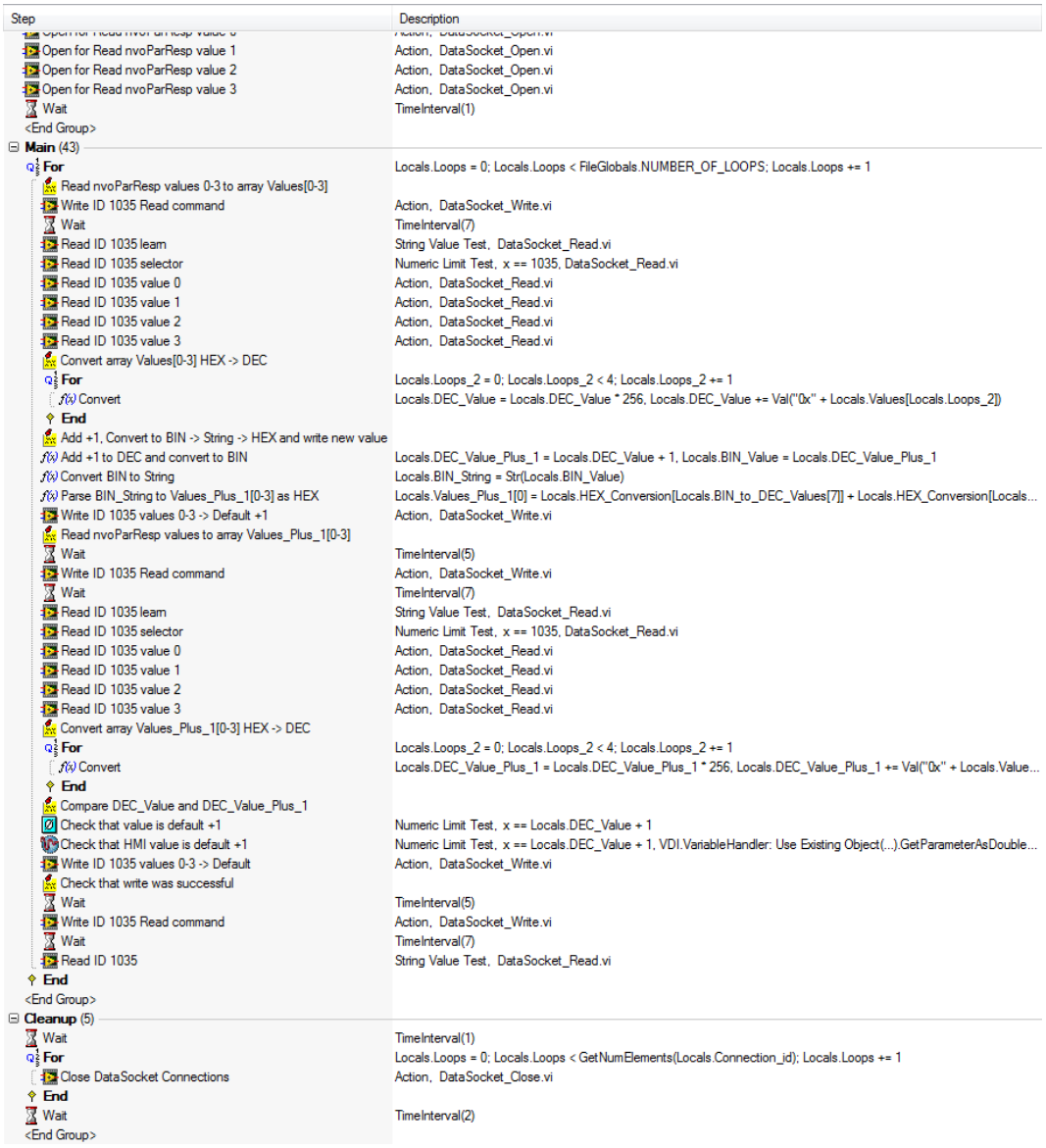
Parametri jakautuu kuuteen osaan, jotka näkyvät taulukossa 1. Jokaisella osalla, sekä parametrillä kokonaisuudessaan on omat osoitteensa OPC-palvelimella, joten voin halutessani kirjoittaa tai lukea koko parametrin sisällön, tai vaikka esimerkiksi heksaluvun kolmannen arvon. Toteutus TestStandiin mutkistui juuri siksi, että luku on heksamuodossa jaettuna neljään osaan.

Taulukko 1. Parametrikomennon sisältö.

Luku tai kirjoitus	LN_LEARN_CURRENT = kirjoitus LN_RECALL = luku
ID	Esim. 1035
Arvot 1–4	Heksaluvun osa: 00 – FF Muodostavat yhdessä heksaluvun: 00000000 – FFFFFFFF

Testitapaus toteutettiin seuraavalla tavalla:

1. Avataan DataSocket-yhteydet.
2. Annetaan parametrin lukukomento ID:lle 1035 ja tarkistetaan, että se meni perille.
3. Luetaan arvot 1–4 erikseen taulukkoon.
4. Muutetaan arvot desimaaliluvuiksi järjestyksessä, kerrotaan aina 256:lla ja lisätään edelliseen arvoon. Näin saadaan muutettua koko heksaluku desimaalimuotoon.
5. Lisätään saatuun desimaalilukuun 1.
6. Muutetaan desimaaliluku 32-bittiseksi binääriluvuksi, koska jos muutetaan suoraan heksamuotoon, edeltävät nollat jäävät pois.
7. Muutetaan binääriluku string-muotoon
8. Otetaan parsella stringistä neljä merkkiä kerrallaan, muutetaan desimaalien kautta heksaluvuksi ja tallennetaan taulukkoon niin, että samaan soluun tulee aina kaksi peräkkäistä heksalukua yhdistettynä. Nyt alkuperäiseen luettuun arvoon on lisätty 1 ja se on taulukossa oikeassa muodossa jaettuna neljään osaan kirjoitusta varten.
9. Kirjoitetaan parametri kokonaisuudessaan uusilla arvoilla ID:lle 1035.
10. Toistetaan kohdat 1–4, eli luetaan arvot ja tehdään niistä desimaaliluku.
11. Tarkistetaan, että luettu arvo on alkuperäinen arvo lisättynä yhdellä.
12. Tarkistetaan arvo myös lukemalla se HMI:n kautta.
13. Kirjoitetaan parametri kokonaisuudessaan alkuperäisillä arvoilla ID:lle 1035.
14. Annetaan lukukomento ID:lle 1035 ja tarkistetaan, että se meni perille.
15. Tarkistetaan, että arvot ovat alkuperäiset.
16. Suljetaan DataSocket-yhteydet



Step	Description
Open for Read nvoParResp value 1	Action, DataSocket_Open.vi
Open for Read nvoParResp value 2	Action, DataSocket_Open.vi
Open for Read nvoParResp value 3	Action, DataSocket_Open.vi
Wait	TimeInterval(1)
<End Group>	
Main (43)	
For	Locals.Loops = 0; Locals.Loops < FileGlobals.NUMBER_OF_LOOPS; Locals.Loops += 1
Read nvoParResp values 0-3 to array Values[0-3]	
Write ID 1035 Read command	Action, DataSocket_Write.vi
Wait	TimeInterval(7)
Read ID 1035 learn	String Value Test, DataSocket_Read.vi
Read ID 1035 selector	Numeric Limit Test, x == 1035, DataSocket_Read.vi
Read ID 1035 value 0	Action, DataSocket_Read.vi
Read ID 1035 value 1	Action, DataSocket_Read.vi
Read ID 1035 value 2	Action, DataSocket_Read.vi
Read ID 1035 value 3	Action, DataSocket_Read.vi
Convert array Values[0-3] HEX -> DEC	
For	Locals.Loops_2 = 0; Locals.Loops_2 < 4; Locals.Loops_2 += 1
f(x) Convert	Locals.DEC_Value = Locals.DEC_Value * 256, Locals.DEC_Value += Val("0x" + Locals.Values[Locals.Loops_2])
End	
Add +1, Convert to BIN -> String -> HEX and write new value	
f(x) Add +1 to DEC and convert to BIN	Locals.DEC_Value_Plus_1 = Locals.DEC_Value + 1, Locals.BIN_Value = Locals.DEC_Value_Plus_1
f(x) Convert BIN to String	Locals.BIN_String = Str(Locals.BIN_Value)
f(x) Parse BIN_String to Values_Plus_1[0-3] as HEX	Locals.Values_Plus_1[0] = Locals.HEX_Conversion[Locals.BIN_to_DEC_Values[7]] + Locals.HEX_Conversion[Locals...
Write ID 1035 values 0-3 -> Default +1	Action, DataSocket_Write.vi
Read nvoParResp values to array Values_Plus_1[0-3]	
Wait	TimeInterval(5)
Write ID 1035 Read command	Action, DataSocket_Write.vi
Wait	TimeInterval(7)
Read ID 1035 learn	String Value Test, DataSocket_Read.vi
Read ID 1035 selector	Numeric Limit Test, x == 1035, DataSocket_Read.vi
Read ID 1035 value 0	Action, DataSocket_Read.vi
Read ID 1035 value 1	Action, DataSocket_Read.vi
Read ID 1035 value 2	Action, DataSocket_Read.vi
Read ID 1035 value 3	Action, DataSocket_Read.vi
Convert array Values_Plus_1[0-3] HEX -> DEC	
For	Locals.Loops_2 = 0; Locals.Loops_2 < 4; Locals.Loops_2 += 1
f(x) Convert	Locals.DEC_Value_Plus_1 = Locals.DEC_Value_Plus_1 * 256, Locals.DEC_Value_Plus_1 += Val("0x" + Locals.Value...
End	
Compare DEC_Value and DEC_Value_Plus_1	
Check that value is default +1	Numeric Limit Test, x == Locals.DEC_Value + 1
Check that HMI value is default +1	Numeric Limit Test, x == Locals.DEC_Value + 1, VDI.VariableHandler: Use Existing Object(...).GetParameterAsDouble...
Write ID 1035 values 0-3 -> Default	Action, DataSocket_Write.vi
Check that write was successful	
Wait	TimeInterval(5)
Write ID 1035 Read command	Action, DataSocket_Write.vi
Wait	TimeInterval(7)
Read ID 1035	String Value Test, DataSocket_Read.vi
End	
<End Group>	
Cleanup (5)	
Wait	TimeInterval(1)
For	Locals.Loops = 0; Locals.Loops < GetNumElements(Locals.Connection_id); Locals.Loops += 1
Close DataSocket Connections	Action, DataSocket_Close.vi
End	
Wait	TimeInterval(2)
<End Group>	

Kuva 13. Parameter Read/Write Test (ID 1035) toteutettuna TestStandiin.

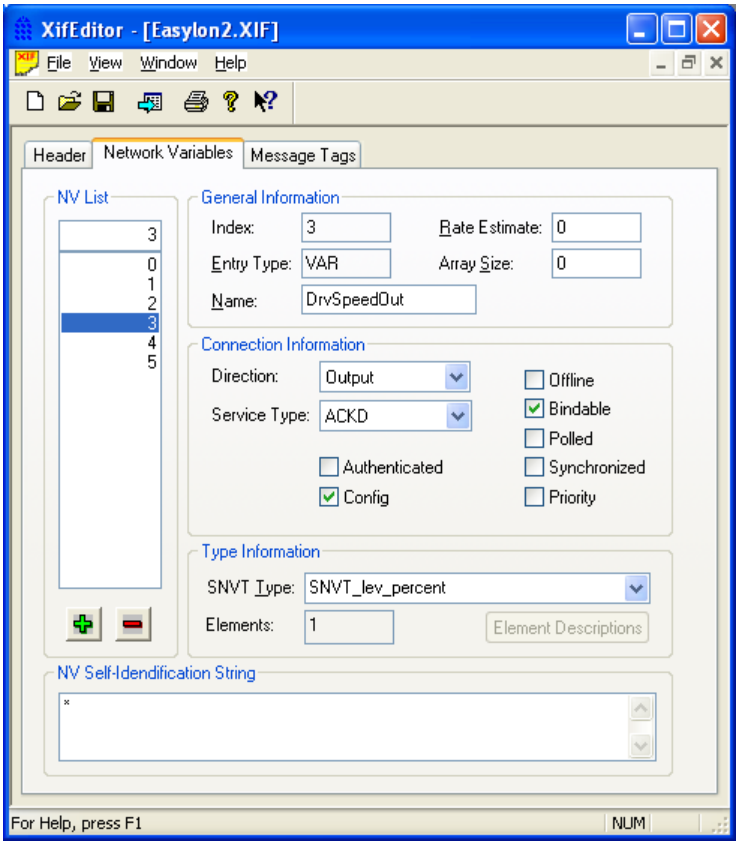
Testitapaukset rakennettiin yksi kerrallaan järjestyksessä omiin sekvensseihinsä. Jokaisen testitapauksen valmistuttua sen toiminta testattiin ensin yksin, ja aina kun löytyi ongelma, se korjattiin ja testi ajettiin uudelleen. Usein ongelmat olivat ajoituksissa, kun kirjoitus- ja lukuoperaatiot tapahtuivat peräkkäin, eikä NV:n arvo OPC-palvelimella ollut ehtinyt päivittymään. Kun testitapaus oli mennyt läpi useita kertoja ja toimi halutulla tavalla, testattiin sitä yhdessä muiden kanssa samalla periaatteella. Välillä ongelmat löytyivät vasta kun muutkin testitapaukset olivat mukana, jos esimerkiksi jokin NV oli jäänyt palauttamatta oletusarvoonsa edellisen sekvenssin jälkeen.

Kun kaikki muut testitapaukset toimivat oikein, oli aika miettiä send heartbeatia. Send heartbeat määrittää maksimiajan, jonka jälkeen kolme NV:tä päivittyy automaattisesti. Nämä NV:t ovat:

- nvoDrvCurnt
- nvoDrvSpeed
- nvoDrvPwr

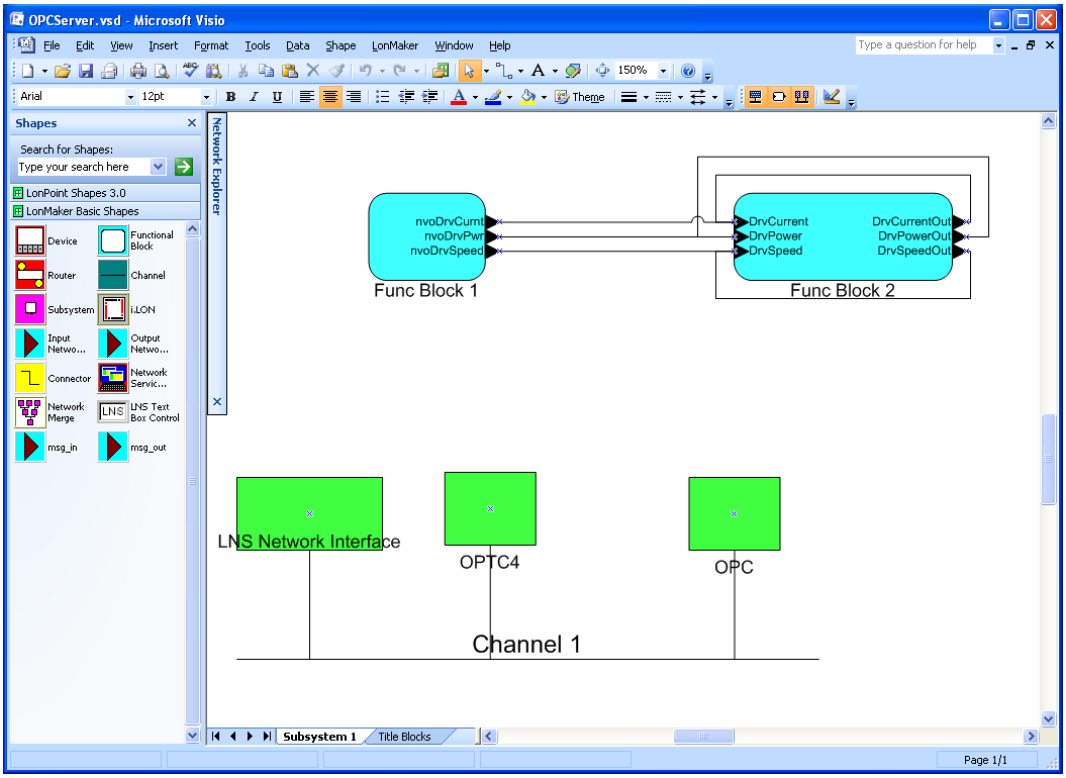
Kun send hearbeatin arvoksi annetaan 10, nämä kolme päivittyvät aina, kun niitä luetaan, tai kymmenen sekunnin välein. OPC-palvelimelle pitää luoda uusi solmu, jossa on kolme uutta NV:tä ja sitoa ne alkuperäisiin, jolloin niiden arvot päivittyvät yhtä aikaa alkuperäisten kanssa. Uusista muuttujista voidaan lukea ja niihin voidaan kirjoittaa arvoja, ilman että alkuperäiset muuttujat päivittyvät, näin voidaan testata toimiiko päivitys kymmenen sekunnin välein. Esimerkiksi jos NV:n nvoDrvSpeed arvo on 0, kirjoitetaan siihen sidottuun muuttujaan jokin muu arvo ja tarkistetaan kymmenen sekunnin kuluttua, onko arvo palautunut nollaksi.

Easylon OPC-palvelimen dokumentaation mukaan sitominen pitäisi onnistua, sitä kysyttiin myös Gesyteciltä ja he vahvistivat asian. Toimenpide tapahtui Echelonin LonMaker Integration Toolilla. Aluksi piti kuitenkin luoda xif-tiedosto uusille muuttujille, tämä tehtiin xif-editorilla. Tarvittiin kolme tulo-muuttujaa ja kolme lähtö-muuttujaa.



Kuva 14. Xif-editorilla luotu xif-tiedosto uusille muuttujille.

LonMaker Integration Toolilla luotiin OPT-C4-optiokortin alkuperäisille muuttujille oma lohko, jonka tiedot saatiin optiokortin xif-tiedostosta. Uutta xif-tiedostoa käyttämällä luotiin verkkoon uusi solmu ja uusille muuttujille oma lohko. Alkuperäiset lähtö-muuttujat sidottiin uusiin tulo-muuttujiin vetämällä johdot lohkojen välille. Uudet tulo-muuttujat taas sidottiin uusiin lähtö-muuttujiin saman lohkon sisällä. Tarvittiin kuusi uutta muuttujaa, koska lähtöjä ei voida sitoa suoraan toisiinsa.



Kuva 15. LonMakerin Microsoft Visio -käyttöliittymä. Alkuperäiset NV:t on sidottu uuden solmun muuttujiin.

LonMakerissa on myös ominaisuus, jolla voidaan luoda tietokanta OPC-palvelinta varten. Luotiin uusi tietokanta, mikä sisältää molemmat verkon solmut ja kaikki NV:t. Avattiin OPC-palvelin ja otettiin uusi tietokanta käyttöön. Nyt palvelimella näkyi myös uusi solmu ja muuttujat päivittyivät niin kuin niiden pitikin. Send Hearbeat Test oli tämän jälkeen mahdollista toteuttaa TestStandiin.

Kun kaikki testitapaukset oli tehty, testattiin koko järjestelmää vielä useita kertoja ja tehtiin pieniä muutoksia joihinkin kohtiin. Muutokset kohdistuivat pääosin kommentointiin ja muutenkin selkeyttä pyrittiin parantamaan. Viimeistelyn jälkeen työ oli valmis.

6 YHTEENVETO JA JOHTOPÄÄTÖKSET

Työn aiheena oli automaattisen testausjärjestelmän luominen LonWorks-kenttäväylälle. Testausohjelmistona käytettävä NI TestStand piti saada kommunikoimaan taajuusmuuttajan kanssa, jonka jälkeen oli vuorossa testitapausten ohjelmointi LonWorksin testispesifikaatiota noudattaen.

Määrittely- ja suunnitteluvaiheessa keskustelin aiheesta muun henkilöstön kanssa, sekä etsin tietoa internetistä ja olin yhteydessä suoraan valmistajiin, kunnes työssä käytettävät laitteisto ja ohjelmisto olivat selvillä, sekä toteutussuunnitelma laadittu. Lähtökohdat olivat haastavat, koska minulla ei ollut aikaisempaa kokemusta National Instrumentsin ohjelmistoista, eikä OPC-palvelimista. Koulussa oli käsitelty kenttäväylistä Modbusia, mutta LonWorks oli uusi tuttavuus. Vaconin järjestämä kurssi auttoi hyvin alkuun TestStandin ja LabVIEW:n kanssa ja työn edetessä opin jatkuvasti uutta niistä, sekä yleensä testausjärjestelmistä ja kenttäväylistä.

Testausympäristön rakentaminen onnistui hyvin ja tässä vaiheessa alun perin suunnitellusta aikataulusta oltiin jopa hieman edellä. Testitapausten luominen TestStandiin osoittautui välillä haastavaksi, vaikka ne olivat hyvin kuvattuina testispesifikaatiossa. Testitapausten ohjelmoinnissa suuri osa ajasta meni myös testien ajamiseen vikojen korjaamisen tai muiden muutosten jälkeen. Ongelmia aiheutti varsinkin viallinen ohjauskortti testilaitteena toimineessa taajuusmuuttajassa, jonka takia testit epäonnistuivat joskus satunnaisesti. Ajattelin vian olevan ohjelmoinnissa tai ohjelmistoissa, mutta ongelman oikea syy selvisi, kun taajuusmuuttajan tehdasasetusten palautuskin alkoi oikkuilla. Ohjauskortin vaihto poisti kaikki ongelmat. Tämä viivytti työn valmistumista jonkin verran, mutta alkuperäisessä aikataulussa pysyttiin silti kohtuullisen hyvin.

Työn lopputulos oli tavoitteiden mukainen ja automaattinen testausjärjestelmä toimii odotetulla tavalla. Testausraportit generoituvat automaattisesti ja tulokset ovat niissä selkeästi esillä. Testitapaukset katselmoitiin Vaconilla ja niiden todettiin olevan testispesifikaation mukaisia. Pyrin ohjelmoimaan ja kommentoimaan testitapaukset selkeästi niin, että niistä on helppo seurata, missä kohtaa testispesifikaatiota mennään. Onnistuin tässä Vaconilta saadun palautteen perusteella

kohtalaisen hyvin. Selkeys ja modulaarisuus ovat tärkeitä siinä vaiheessa, kun testispesifikaatiota päivitetään ja testitapauksia joudutaan muokkaamaan. Tätä kirjoittaessa automaattista testausjärjestelmää ollaan ottamassa, tai se on jo otettu vakituiseen käyttöön. Sillä pystytään korvaamaan täysin LonWorksin manuaalinen testaus.

Vaconilla ohjaajanani toiminut Jari Hill oli suurena apuna koko projektin ajan ja auttoi aina ongelmatilanteissa. Varsinkin alkuvaiheessa opastusta tarvittiin, kun uutta asiaa tuli paljon.

Automaattinen testaus on merkittävä edistysaskel manuaaliseen testaukseen verrattuna. Järjestelmän rakentaminen vie aikaa ja resursseja, mutta projektin kannattavuuden määrittelee se, kuinka usein testejä on tarvetta ajaa. Tämä pitää aina miettiä tapauskohtaisesti ja Vaconilla oli tälle selkeä tarve. Kun järjestelmä on kerran tehty kunnolla modulaarisesti ja selkeästi, sitä on myös helppo jatkossa päivittää vastaamaan uusia testispesifikaation versioita.

LonWorksin tapauksessa automaattinen testausjärjestelmä ehtii ajamaan testit läpi kymmeniä kertoja siinä ajassa, kun ne ehditään manuaalisesti käymään kerran läpi. Suurin etu on kuitenkin se, että testejä voidaan ajaa vaikka jatkuvasti, jos tarvetta on, muiden töiden siitä kärsimättä. Testaaja voi käynnistää testit etänä vaikka palaverista tai kotoaan, ja ne pyörivät itsenäisesti niin monta kertaa kuin on määrätty. Tulokset ovat luotettavampia, koska testejä ehditään ajamaan enemmän ja inhimillisten virheiden mahdollisuus vähenee huomattavasti.

National Instrumentsin ohjelmistot tulivat tutuksi työtä tehdessä, TestStandin käyttö onnistuu jo kohtuullisen hyvin ja se vaikutti todella joustavalta ympäristöltä. Graafinen ohjelmointi LabVIEW:lla oli myös uutta ja mielenkiintoista, siihenkin sain hyvän pohjan. Olen suuntautunut opinnoissani sulautettuihin järjestelmiin ja tietoliikennetekniikkaan, mielestäni opinnäytetyön aihe oli sopiva ja työ täydensi hyvin jo koulussa oppimaani.

LÄHDELUETTELO

/1/ Vacon Oyj 2011, Vacon Oyj - Yleistä [viitattu 31.1.2011]. Saatavilla internetissä: http://www.vacon.fi/Default.aspx?id=461919

/2/ Vacon Oyj 2011, Vacon variable speed AC drives – Products – What is an AC drive? [viitattu 1.2.2011]. Saatavilla internetissä: http://www.vacon.com/Default.aspx?id=464743

/3/ Emerson Indrustial Automation 2009, A Guide to Popular Fieldbus Systems for use with Variable Speed Drives [viitattu 4.2.2011]. Saatavilla internetissä: http://www.driveka.ru/upload/iblock/fc8/fieldbus%20iss2%20low%20res.pdf

/4/ Vacon Oyj 2001, Fieldbus technology provides reliable and smooth process control [viitattu 4.2.2011]. Saatavilla internetissä: http://www.vacon.com/Default.aspx?Id=461215

/5/ The Modbus Organization 2011, Modbus FAQ: About The Modbus Organization [viitattu 8.2.2011]. Saatavilla internetissä: http://www.modbus.org/faq.php

/6/ Lammert Bies 2010, Modbus interface tutorial [viitattu 8.2.2011]. Saatavilla internetissä: http://www.lammertbies.nl/comm/info/modbus.html

/7/ Weighing-systems.com, Fieldbus overview [viitattu 11.2.2011]. Saatavilla internetissä: http://www.weighing-systems.com/TechnologyCentre/fieldbus1.html

/8/ Sensor-Technik Wiedemann GMBH 2011, SafetyCAN [viitattu 13.2.2011]. Saatavilla internetissä: http://www.sensor-technik.de/en/software/protocols/safteycan

/9/ EMS Thomas Wünsche 2009, EMS - Fieldbus CAN, Usage areas, Controlling, Sensors, actuators ... [viitattu 13.2.2011]. Saatavilla internetissä: http://www.ems-wuensche.com/about-can/can-usageareas-info.html

/10/ PI - PROFIBUS & PROFINET International, Technology [viitattu 14.2.2011]. Saatavilla internetissä: http://www.profibus.com/technology/

/11/ PI - PROFIBUS & PROFINET International, For FA [viitattu 14.2.2011]. Saatavilla internetissä: http://www.profibus.com/technology/profibus/for-fa/

/12/ PI - PROFIBUS & PROFINET International, For PA [viitattu 14.2.2011]. Saatavilla internetissä: http://www.profibus.com/technology/profibus/for-pa/

/13/ CAN in Automation (CiA), CAN history [viitattu 14.2.2011]. Saatavilla internetissä: http://www.can-cia.de/index.php?id=161

/14/ Actel 2010, CAN/CANbus and CAN Protocol Licensing [viitattu 14.2.2011]. Saatavilla internetissä: http://www.actel.com/ipdocs/CANbus_lic_RS.pdf

/15/ Greentech Media 2009, Echelon Beefs Up LonWorks [viitattu 16.2.2011]. Saatavilla internetissä: http://www.greentechmedia.com/articles/read/echelon-beefs-up-lonworks-5814/

/16/ Anybus 2010, Lonworks fieldbus network - technology overview [viitattu 16.2.2011]. Saatavilla internetissä: http://www.anybus.com/technologies/lonworks.shtml

/17/ HVAC Airconditioning and Heating 2006, What Is HVAC? [viitattu 16.2.2011]. Saatavilla internetissä: http://www.hvachome.net/

/18/ Echelon 2011, LonWorks FAQ [viitattu 16.2.2011]. Saatavilla internetissä: http://www.echelon.com/communities/developers/lonworks/faq.htm

/19/ Real Time Automation 2000, LonWorks Protocol Overview [viitattu 16.2.2011]. Saatavilla internetissä: http://www.rtaautomation.com/lonworks/

/20/ Echelon 2011, LNS Network Operating System [viitattu 16.2.2011]. Saatavilla internetissä: http://www.echelon.com/products/development/lns/default.htm

/21/ SmartBear Software 2011, Regression Testing [viitattu 21.2.2011]. Saatavilla internetissä: http://www.automatedqa.com/techpapers/testcomplete/regression-testing/

/22/ Vacon Oyj, LonWorks Option Board OPT-C4 User's Manual [viitattu 4.3.2011]. Saatavilla internetissä: http://www.vacon.com/File.aspx?id=462930&ext=pdf&routing=396771&name=UD00887C

/23/ Gesytec GmbH, Easylon VNI Interface USB [viitattu 5.3.2011]. Saatavilla internetissä: http://www.gesytec.de/uk/produkte/easylon-for-lonworks/lon-interfaces/easylon-vni-interface-usb/

/24/ OPC Programmers' Connection 2010, History of OPC [viitattu 7.3.2011]. Saatavilla internetissä: http://www.opcconnect.com/history.php

/25/ MatrikonOPC 2011, OPC Server [viitattu 7.3.2011]. Saatavilla internetissä: http://www.matrikonopc.com/opc-server/index.aspx

/26/ MatrikonOPC 2011, OPC Data Access (OPC DA) Versions & Compatibility [viitattu 7.3.2011]. Saatavilla internetissä: http://www.matrikonopc.com/opc-server/opc-data-access-versions.aspx

/27/ National Instruments 2011, DataSocket Technical Information [viitattu 8.3.2011]. Saatavilla internetissä: http://www.ni.com/datasocket/ds_what.htm

/28/ National Instruments 2011, NI LabVIEW Supports Multiple Targets and Operating Systems [viitattu 8.3.2011]. Saatavilla internetissä: http://www.ni.com/labview/whatis/multiple-targets/

/29/ Echelon Corporation 2011, LonMaker® Integration Tool, Turbo Edition [viitattu 11.3.2011]. Saatavilla internetissä: http://www.echelon.com/products/networktools/lonmaker/default.htm

/30/ LonMark 2011, Guides & Specifications - Functional Profiles [viitattu 14.3.2011]. Saatavilla internetissä: http://www.lonmark.org/technical_resources/guidelines/docs/profiles/

/31/ Software Testing Help 2007, Black Box Testing: Types and techniques of BBT [viitattu 30.3.2011]. Saatavilla internetissä: http://www.softwaretestinghelp.com/black-box-testing/

/32/ Software Testing Fundamentals, Unit Testing [viitattu 4.4.2011]. Saatavilla internetissä: http://softwaretestingfundamentals.com/unit-testing/

/33/ Software Testing Fundamentals, Integration Testing [viitattu 4.4.2011]. Saatavilla internetissä: http://softwaretestingfundamentals.com /integration-testing/

/34/ Software Testing Fundamentals, System Testing [viitattu 4.4.2011]. Saatavilla internetissä: http://softwaretestingfundamentals.com/system-testing/

/35/ Beizer, Boris (1990). Software Testing Techniques, 2. painos. New York: Van Nostrand Reinhold.

/36/ Software Testing Help 2008, What is Boundary value analysis and Equivalence partitioning? [viitattu 10.4.2011]. Saatavilla internetissä: http://www.softwaretestinghelp.com /what-is-boundary-value-analysis-and-equivalence-partitioning/

/37/ Software Testing Help 2007, How to write effective Test cases [viitattu 10.4.2011]. Saatavilla internetissä: http://www.softwaretestinghelp.com /how-to-write-effective-test-cases-test-cases-procedures-and-definitions/